DEPARTMENT OF COMPUTER SCIENCE

Mobility Problems in Distributed Search and Combinatorial Games

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy by

Ioannis Lamprou

November 2018

Abstract

Mobility Problems in Distributed Search and Combinatorial Games

by Ioannis Lamprou

This thesis examines a collection of topics under the general notion of *mobility of agents*. We examine problems where a set of entities, perceived as robots or tokens, navigate in some given (discrete or continuous) environment to accomplish a goal. The problems we consider fall under two main research fields. First, *Distributed Search* where the agents cooperate to explore their environment or search for a specific target location within it. Second, *Combinatorial Games*, in the spirit of *Pursuit-Evasion*, where the agents are now divided into two groups with complementary objectives competing against each other. More specifically, we consider three distinct problems: disk evacuation, exploration of dynamic graphs and eternal domination.

In *Disk Evacuation*, two robots with different speeds aim to discover an unknown exit lying on the boundary of a unit disk. For a wide range of speeds, we provide matching upper and lower bounds. In *Dynamic Graph Exploration*, we analyze the exploration time for a randomly-walking agent wishing to visit all the vertices of a stochastically-evolving graph. In *Eternal Domination*, we consider rectangular grid graphs and upper bound the amount of guard agents needed to perpetually defend the vertices against an attacker.

To my family

Konstantinos, Anna, Georgios

“ Ὣς οὐδὲν γλύκιον ἧς πατρίδος οὐδὲ τοκήων γίγνεται ”

(Ομήρου Οδύσσεια, ι΄ 34)

“ Nothing is sweeter than a man’s homeland and his parents ”

(Homer, Odyssey, 9:34)

Acknowledgements

I would like to express my gratitude to my PhD supervisors Russell Martin and Sven Schewe for their time, feedback, and friendship during the last three years. This thesis is a direct product of our cooperation. Producing it would not have been possible without them.

I would like to thank my coauthor, Paul Spirakis, for his cooperation, willingness to work with me, and his overall guidance with respect to academic matters. Also, I thank my colleagues from the University of Athens, Ioannis Sigalas and Vassilis Zissimopoulos, for their constant support and collaboration.

Many thanks go to my academic advisors, Leszek Gąsieniec and Dariusz Kowalski, who monitored my progress and provided me with useful feedback and research suggestions.

I am grateful to the Department of Computer Science, University of Liverpool, for funding my studies with a graduate teaching assistantship. Thanks also to the Network and Sciences Technologies (NeST) initiative, School of Electrical Engineering, Electronics and Computer Science, University of Liverpool, for partially supporting my research expenses.

I would like to acknowledge all my colleagues and friends at the University of Liverpool who contributed to lively aspects of this three-year PhD experience.

Last but not least, I owe a big thank you to my family and close friends, whose love and selfless support extend far beyond my time period as a PhD candidate.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

FES	Fast-Explores Strategies
SES	Slow-Explores Strategies
BES	Both-Explore Strategies
BSP	Both to the Same Point
RWD	Random Walk with a Delay
RWA	Random Walk on what's Available
EUE	Edge-Uniform Evolution
SRW	Simple Random Walk
LWD	Lower Walk with a Delay
UWD	Upper Walk with a Delay

Chapter 1

Introduction

Mobility is a pivotal notion in today's world: The technological advancements in the last century have provided the framework for human and, most importantly, non-human entities, also called *(mobile) agents* (e.g., software or robots), to engage in a plethora of scenarios. The agents have a certain set of capabilities and they move over time and through space in order to accomplish a designated goal. Mobility problems are ubiquitous: Nowadays, easy to obtain low-cost machines, each with very limited functionality, can nonetheless be powerful enough to perform complex tasks when put together. Applications arise in areas as diverse as network security [20] and cave exploration [24], web caching [59] and search-and-rescue [77], and, distributed networking [92] and disaster mitigation [57].

However, computing such moving schemes of good quality, in terms of time, space or other resources used, may be a mission of significant difficulty due to various (and possibly conflicting) requirements. It is this problem that *Theoretical Computer Science* aims to resolve. That is, we seek to provide optimal and well-designed *strategies* in order for the agents to perform efficiently their respective goal. To do so, we employ techniques from the fields of *Algorithm Design* and *Distributed Computing*. Depending on the task in question, a rigorous analysis of such strategies may be based on several mathematical tools like Discrete/Continuous Optimization, Graph Theory, Geometry, and others. Other features, like the environment or the entities' capabilities, may also have crucial impact on the proposed strategies. Overall, the design and analysis of such routines is proven to be a multifaceted research direction of ever-growing importance due to advancements in technology and networking leaning toward more and more distributed settings.

In the context of this thesis, the research interest lies in optimization settings where agents navigate within a certain environment and either cooperate or fight against each other to perform a certain task. The research aspect for such problems is twofold. That is, we wish to devise strategies enabling the agents to perform their task efficiently and, at the same time, we seek to produce results correlating task efficacy to properties of the environment. The focus of this thesis is on problems arising in two main areas, namely *Distributed Search* and *Combinatorial Games*.

1.1 Distributed Search

Imagine a scenario where a set of agents is initially placed in a specific spatial environment. The latter could be either *discrete*, e.g., a graph in the sense that it represents a network topology or *continuous*, e.g., a subspace of Euclidean space, in the sense that it depicts an actual geographical setting. The agents need to perform a certain *search* task within their environment. Examples of such tasks include searching for a specific target, discovering some part of or the whole environment, meeting at a specific place, that is, arranging a rendezvous, and agreeing on a property of the environment.

The topology itself is merely one obstacle toward the efficient completion of a task. The way the agents *communicate* with each other is another. Communication settings include *wireless* communication, where the agents can contact each other at any time, *face-to-face* communication, where the agents can only interact when they physically meet in the environment, and finally settings where an agent can *leave a note* (sometimes called *pebble* [16]) for another agent to read when it arrives there. Different communication protocols may, indeed, significantly affect the complexity of resolving a task.

Further crucial attributes for the performance of a task comprise the (possibly limited) memory of the agents, their speed, e.g., it might not be the same for every agent, and dynamic changes to the environment in which the task takes place, e.g., a task taking place in a *temporal network* [96].

The aforementioned set of problems is mostly met in literature under the *(Distributed) Search Problems* theme. The research objective is to design efficient, optimal, *(distributed) algorithms*, which formally define the actions followed by each agent to perform the task in question. A seminal paper on a search problem for the continuous case can be found in [8]; see [94] for a seminal example on the discrete case. In books [4, 5], various search problems are surveyed in a game theoretic perspective.

In this thesis, we examine two problems in the above context. In Chapter 2, we study a disk evacuation setting [86], where two robots with different speeds search for an unknown exit on the boundary of a disk. In Chapter 3, we consider an exploration situation [90], where a single randomly-moving robot traverses a stochastically-evolving dynamic graph.

1.2 Combinatorial Games

Let us now consider another setting where the set of agents is partitioned into two opposing teams playing against each other in rounds. A team of k agents and another of a sole agent. The former seek to capture the latter, while the latter strives to perpetually evade this outcome. The just sketched *Cops & Robber* game [19] is a classic *combinatorial game* of which many variants have been surveyed. A first result was the characterization of 1-cop winning graphs, independently made by Nowakowski and Winkler [99], and Quillot [102]. Aigner and Fromme [1] proved that three cops suffice to capture the robber in any planar

graph. Meyniel conjectured that roughly $\sqrt{n}$ guards suffice to capture the robber in any given graph with n vertices. The conjecture remains widely open for almost forty years, despite some relatively recent progress [93, 105].

In general, *Combinatorial Game Theory* [2] is interested in games where two players play alternately and both enjoy perfect information over the game. At each turn, each player is familiar with the current and all previous game configurations and picks an action out of a set of publicly known predefined actions. Nevertheless, visibility-less versions have also been studied: for an example see [75]. Furthermore, the games are deterministic in the sense that no action performed is dependent on any source of randomness. By the end of the game, one player wins and the other loses; there may be no tie. Our main interest lies in the area of *Pursuit-Evasion Games* [100], where token mobility is a cornerstone for the analysis. Similarly to distributed search, the environment of the game is critical toward determining its key features: there is a great amount of literature in the continuous, e.g., starting from Littlewood's *lion and man* problem [18], as well as in the discrete case [60]. Further, for applications in the area of mobile robotics, see [35].

In Chapter 4, we examine such a game on the topic of graph protection [87]. The motivation for studying such a game goes back to the Roman Empire and the defence strategies of Emperor Constantine I [103, 108] and is applicable to similar military policies nowadays. In the *Eternal Domination Game* [28], a set of tokens, the guards, lie on a graph, while an invisible rioter attacks a single vertex in each turn. Guards win if at least one of them can always immediately visit a just attacked vertex. The rioter wins, otherwise.

1.3 Preliminaries

1.3.1 Graph Theory

We now provide some preliminary graph theory notions that we later use in Chapters 3,4.

Let $G = (V(G), E(G))$ be a simple connected undirected graph. We denote an edge between two connected vertices, namely v and u, as $(u, v) \in E(G)$, or equivalently (v, u).

The *open neighborhood* of a subset of vertices $S \subseteq V(G)$ is defined as $N(S) = \{v \in V(G) \setminus S : \exists u \in S \text{ such that } (u, v) \in E(G)\}$ and the *closed neighborhood* as $N[S] = S \cup N(S)$. For a single vertex $v \in V(G)$, we simplify the notation $N(\{v\})$ to $N(v)$ and, similarly, $N[\{v\}]$ to $N[v]$. By $d(v)$, we denote the degree, i.e., the number of neighbors, of a vertex $v \in V$. In other words, $d(v) = |N(v)|$.

A *path* of length $n-1 \in \mathbb{N}$, namely P_n, is a graph where $V(P_n) = \{v_0, v_1, \ldots, v_{n-1}\}$ and $E(P_n) = \{(v_0, v_1), (v_1, v_2), \ldots, (v_{n-2}, v_{n-1})\}$. The *Cartesian product* of two graphs G and H is another graph denoted $G \square H$ where $V(G \square H) = V(G) \times V(H)$ and two vertices (v, v') and (u, u') are adjacent if either $v = u$ and $(v', u') \in E(H)$ or $v' = u'$ and $(v, u) \in E(G)$. A *grid*, namely $P_m \square P_n$, is the Cartesian product of two paths of lengths $m-1, n-1 \in \mathbb{N}$.

A subset of vertices for which all possible edges are present is called a *clique*.

A set of vertices $S \subseteq V(G)$ is called a *dominating set* of G if $N[S] = V(G)$. That is, for each $v \in V(G)$, either $v \in S$ or there exists a vertex $u \in S$ ($u \neq v$) such that $(u, v) \in E(G)$. A minimum-size such set, say S^*, is called a *minimum dominating set* of G and $\gamma(G) = |S^*|$ is defined as the *domination number* of G. For grids, we simplify $\gamma(P_m \square P_n)$ to $\gamma_{m,n}$.

A lollipop graph L_n^k consists of a clique on k vertices and a path on $n - k$ vertices connected with a cut-edge, i.e., an edge whose deletion makes the graph disconnected.

For any disambiguation or further information on basic graph-theoretic notions, the reader is referred to a graph theory textbook, e.g., [21, 112].

1.3.2 Random Walks

Let us hereby define a few standard notions related to a simple random walk performed by a single agent on a simple connected graph $G = (V, E)$ (for use in Chapter 3).

A simple random walk is a Markov chain where, for $v, u \in V$, we set $p_{vu} = 1/d(v)$, if $(v, u) \in E$, and $p_{vu} = 0$, otherwise. That is, an agent performing the walk chooses the next vertex to visit uniformly at random amongst the set of neighbors of its current vertex. Given two vertices v, u, the expected time for a random walk starting from v to arrive at u is called the *hitting time* from v to u and is denoted by H_{vu}. The *cover time* of a random walk is the expected time until the agent has visited each vertex of the graph at least once.

Let P stand for the stochastic matrix describing the transition probabilities for a random walk (or, in general, a discrete-time Markov chain) where p_{ij} denotes the probability of transition from vertex i to vertex j, $p_{ij} \geq 0$ for all i, j and $\sum_j p_{ij} = 1$ for all i. Then, the matrix P^t consists of the transition probabilities to move from one vertex to another after t time steps and we denote the corresponding entries as $p_{ij}^{(t)}$. Asymptotically, $\lim_{t\to\infty} P^t$ is referred to as the *limiting distribution* of P.

For a general introduction to Markov chains, we cite textbooks [68, 98].

1.4 Summary of Results

In the paragraphs below, we provide a short summary for each topic examined in this thesis.

Disk Evacuation. In the fast evacuation problem, we study the path planning problem for two robots who want to minimize the worst-case evacuation time on the unit disk, that is, the time till both of them evacuate. The robots are initially placed at the center of the disk. In order to evacuate, they need to reach an unknown point, the exit, on the boundary of the disk. Once one of the robots finds the exit, it will instantaneously, i.e., using wireless communication, notify the other agent who will then follow a straight line to it.

The problem has been studied for robots with the same speed [42]. We study a more general case where one robot has speed 1 and the other has speed $s \geq 1$. We provide optimal evacuation strategies in the case that $s \geq c_{2.75} \approx 2.75$ by showing matching upper

and lower bounds on the worst-case evacuation time. For $1 \leq s < c_{2.75}$, we show (non-matching) upper and lower bounds on the evacuation time with a ratio less than 1.22. Moreover, we demonstrate that a different-speeds generalization of the two-robot search strategy from [42] is outperformed by our proposed strategies for any $s \geq c_{1.71} \approx 1.71$.

The above work is presented in Chapter 2. An extended abstract appeared at the *30th International Symposium on Distributed Computing* (DISC 2016) [86] coauthored with Russell Martin and Sven Schewe. A full version has been submitted to *Theoretical Computer Science*.

Dynamic Graph Exploration. We define a general model of stochastically-evolving graphs, namely the *Edge-Uniform Stochastically-Evolving Graphs*. In this model, each possible edge of an underlying general static graph evolves independently being either alive or dead at each discrete time step of evolution following a (Markovian) stochastic rule. The stochastic rule is identical for each possible edge and may depend on the past $k \geq 0$ observations of the edge's state.

We examine two kinds of random walks for a single agent taking place in such a dynamic graph: (i) The *Random Walk with a Delay* (*RWD*), where at each step the agent chooses (uniformly at random) an incident possible edge, i.e., an incident edge in the underlying static graph, and then it waits till the edge becomes alive to traverse it. (ii) The more natural *Random Walk on what is Available* (*RWA*) where the agent only looks at alive incident edges at each time step and traverses one of them uniformly at random. Our study is on bounding the *cover time*, i.e., the expected time until each vertex is visited at least once by the agent.

For *RWD*, we provide a first upper bound for the cases $k = 0, 1$ by correlating *RWD* with a simple random walk on a static graph. Moreover, we present a modified electrical network theory capturing the $k = 0$ case.

For *RWA*, we derive some first bounds for the case $k = 0$, by reducing *RWA* to an *RWD*-equivalent walk with a modified delay. Further, we also provide a framework shown to compute the exact value of the cover time for a general family of stochastically-evolving graphs in exponential time.

The above work appears in Chapter 3. An extended abstract was presented at the *19th International Symposium on Stabilization, Safety, and Security of Distributed Systems* (SSS 2017) [90] coauthored with Russell Martin and Paul Spirakis. A journal version was published at *Algorithms* [91].

Eternal Domination. In the m-*Eternal Domination* game, a team of guard tokens at first occupies a dominating set on a graph G. An attacker then picks a vertex without a guard on it and attacks it. The guards defend against the attack: one of them has to

move to the attacked vertex, while each remaining one can choose to move to one of his neighboring vertices. The new guards' placement must again be dominating. This attack-defend procedure continues eternally. The guards win if they can eternally maintain a dominating set against any sequence of attacks, otherwise the attacker wins.

The m-*eternal domination number* for a graph G is the minimum amount of guards such that they win against any attacker strategy in G in the above described *all guards move model*. We study rectangular grids and provide the first known general upper bound on the m-eternal domination number for such graphs. Our novel strategy implements a square rotation principle and eternally dominates $m \times n$ grids by using approximately $\frac{mn}{5}$ guards, which is asymptotically optimal even for ordinary domination.

The above work appears in Chapter 4. An extended abstract appeared at the *10th International Conference on Algorithms and Complexity* (CIAC 2017) [87] coauthored with Russell Martin and Sven Schewe. A full version is now in press at *Theoretical Computer Science* [88].

Ultimately, in Chapter 5, we discuss some concluding remarks on the notion of mobility of tokens. Also, we converse about potentially new lines of research within the context of Distributed Search and Combinatorial Games.

On a side note, in [89], we study an optimization problem, whose motivation also stems from mobility concepts in combinatorial games. This last work is not included in this thesis for cohesion of the subject matter purposes.

Chapter 2

Disk Evacuation

2.1 Introduction

Consider a pair of mobile robots in an environment represented by a circular disk of unit radius. The goal of the robots is to find an *exit*, i.e., a point at an unknown location on the boundary of the disk, and both move to this exit. The exit is only recognized when a robot visits it. The robots' aim is to accomplish this task as quickly as possible. This problem is referred to as the *evacuation problem*. The robots start at the center of the disk and can move with a speed not exceeding their maximum velocity, which may be different from one another. They can coordinate their actions in any manner they like, and can communicate wirelessly (instantaneously).

2.1.1 Related work

Evacuation belongs to the realm of distributed search problems, which have a long history in mathematics, computer science, and operations research, see, e.g., [13–15].

Salient features in search problems include the *environment* (a geometric one or graph-based), *mobility* of the robots (how they are allowed to move), *perception* of and *interaction* with the environment, and their *computational* and *communication abilities*. Model tasks include exploring and mapping an unknown environment or finding a (mobile or immobile) target. Examples include cops and robbers games [19] and pursuit-evasion games [100], the "lost at sea" problem [64], the cow-path problem and the plane-searching problem [8, 22, 76, 78]. Other tasks are rendezvous or gathering of mobile agents [84, 85] and evacuation [33, 42, 47]. (Note that we distinguish between the distributed version of evacuation problems involving a search for an unknown exit, and centralized versions typically modeled as (dynamic) capacitated flow problems on graphs, where the exit is known.) A general survey of search and rendezvous problems can be found in [5]. Another related problem is the task of patrolling or monitoring, i.e., the periodic (re)visitation of (part of) the environment [32, 43, 114].

In most of these settings, the typical cost is the time required to finish the task (in a synchronous environment), or the total distance moved by the robots to finish it (in an asynchronous setting). Patrolling has a different "cost", the time between consecutive visits to any point in the region, the so-called "idle time".

A little explored feature of the robots is their *speed*. Most past work has focused on the case where all robots share the same (maximal) speed. Notable exceptions of which the authors are aware include [33], which considers the evacuation problem on the infinite line with robots with distinct maximal speeds, [43], which introduces a non-intuitive ring patrolling strategy using three robots with distinct maximal speeds, and [53, 74], where the rendezvous problem with different speeds in a cycle is studied. It is this feature, robots with different maximal speeds, that we explore in this paper. Such a feature makes our model more general and applicable to real-life scenarios.

The most relevant line of work explores the evacuation problem in the unit disk with two robots with identical speeds. The wireless communication model is studied in [42], where they provide an optimal evacuation strategy. The face-to-face communication model is examined in [23, 42, 47]. In this case, the strategies provided are nearly optimal, yet exact optimality seems to be very difficult to obtain. Hence, a more recent work [34] turns the attention to average-case analysis and discusses its trade-offs with respect to worst-case.

In recent years, many other variations of the problem have appeared in the literature such as first locating a treasure and then evacuating it via the exit [61, 62], evacuating a designated robot first due to security priorities [45, 46], evacuating via two unknown exits [101], and evacuating in the presence of a faulty robot [44].

2.1.2 Our results

We consider the evacuation problem in the unit disk using two robots with distinct maximal speeds: one with speed 1, the second with speed $s \geq 1$. The robots share a common clock and can communicate instantaneously when they have found the exit (wireless communication) and so can synchronize their behavior in the evacuation procedure. We assume that the robots can measure distances to an arbitrary precision (equivalently, they can measure time to an arbitrary precision), and can vary their speeds as they desire, up to their maximum speed. A necessity for robots to travel with less than optimal speed could emerge if further constraints are added to the model, e.g., communication radius restrictions where a faster robot might need to slow down to remain near a slower robot in order to be able to maintain an open communication channel. Note that, in our bounds to follow, the robots always travel at maximum speed.

We show that, even in the case of two robots, the analysis involved in finding (time) optimal evacuation strategies can become intricate with strategies that depend on the ratio of the fast robot's to the slow robot's maximal speed. For large s, we introduce an efficient and non-obvious search strategy, called the *Half-Chord* Strategy, see Figure 2.1. For small

s, we generalize a strategy from [42], namely the *Both-to-the-Same-Point* Strategy (BSP), where the two robots move to the same point on the boundary and then separately explore the boundary in clockwise and counterclockwise directions to find the exit (Figure 2.4a). For values of $s \geq c_{1.86}$ (with $c_{1.86} \approx 1.856$), we show that BSP is not optimal by demonstrating that the Half-Chord Strategy is superior to it. Moreover, we improve on this with the *Fast-Chord* Strategy (Figure 2.5), which outperforms Half-Chord for $1.71 \approx c_{1.71} < s < c_{2.07} \approx 2.07$. We obtain optimality for all $s \geq c_{2.75} \approx 2.75$, in the wireless setting, as we demonstrate matching upper and lower bounds on the evacuation time. For $s \in (1, c_{2.75})$, we provide lower bounds on the evacuation time that do not match the upper bounds provided by the respective search strategies (BSP for $s < c_{1.71}$, Fast-Chord for $s \in [c_{1.71}, c_{2.07})$, and Half-Chord for $s \geq c_{2.07}$). The worst ratio between our upper and lower bound, 1.22, is realized for $s = c_{1.71}$.

Section 2.2 contains a more formal definition of the problem we consider. Section 2.3 contains our upper bounds on the evacuation time, while Section 2.4 has our lower bounds. Finally, Section 2.5 contains a comparison of all obtained bounds and Section 2.6 concludes with some further work suggestions.

2.2 Problem Definition and Strategy Space

In this section, we define the problem in question and discuss the robots' capabilities. Furthermore, we provide a partition of the strategy space and some observations which will be useful in the bounds to follow.

Definition 1. *An "evacuation strategy" is an algorithm which describes how each robot moves such that both robots have evacuated the disk at the end of its execution. A "main evacuation strategy" is an algorithm which describes each robot moves up and until the point where one of the robots has discovered the exit.*

A main evacuation strategy is simply a description of the movement of both robots that respects their speed limits, such that every point on the boundary is eventually visited by at least one robot. A (full) evacuation strategy includes, for each possible exit point, a description of the movement of the agents after the discovery of the exit point. While this second part makes (full) evacuation strategies an uncountable family of movement descriptions, parameterized by the exit point, the optimal movement in this second part is obviously a straight line to the exit, cf. Remark 1, such that we focus on finding a good main evacuation strategy.

Definition 2 (The Fast Evacuation Problem)**.** *Given a unit disk and two robots starting at its center (the former with maximum speed $s \geq 1$ and the latter with maximum speed 1), provide an evacuation strategy such that* both *robots reach an unknown exit lying on a boundary point of the disk.*

The two robots, called *Fast* and *Slow*, occupy the size of a single point, are allowed to move within the entire unit disk with a speed up to their maximal one, can only identify the exit when they stand on it, and can communicate wirelessly at any time. Notice that a robot is aware whether it assumes the role of Fast or Slow and executes the corresponding part of the evacuation strategy. The robots can recognize whether their current location is an interior or a boundary point, can measure distances and time to an arbitrary precision, and have a common perception of time.

The following remark is a direct consequence of the disk environment and the wireless communication used by the robots.

Remark 1. *In any evacuation strategy, when either robot discovers the exit, the optimal strategy of the other one immediately reduces to following a straight line to the exit.*

We now proceed with identifying key aspects of potential strategies.

Definition 3. *A "both-explore" strategy is a strategy for both robots to evacuate the disk, where (in the worst-case) each of them explores at least two distinct points on the boundary. We define the set of all both-explore strategies as BES.*

Definition 4. *A "fast-explores" strategy is a strategy where* only Fast *explores the boundary searching for the exit. At no time during the execution of the algorithm does Slow visit a point on the boundary other than the exit point. We define the set of all fast-explores strategies as FES.*

Definition 5. *A "slow-explores" strategy is a strategy where* only Slow *explores the boundary searching for the exit. At no time during the execution of the algorithm does Fast visit a point on the boundary other than the exit point. We define the set of all slow-explores strategies as SES.*

Notice that, for $s = 1$, if only one robot explores the boundary, we randomly assign such a strategy to FES or SES. Below, let ALL stand for the set of all evacuating strategies.

Proposition 1. *(BES, FES, SES) forms a partition of ALL.*

Proof. BES, FES and SES are pairwise disjoint, since if only Fast or Slow explores, then both do not explore, and if only Fast explores, then Slow does not, and vice versa.

$ALL = BES \cup FES \cup SES$, since for any possible strategy at least one robot explores the boundary. □

We remark that, when considering SES and FES strategies, it can become a burden to forcefully keep the non-exploring robot away from the boundary. For example, if we only want Slow to explore in an SES strategy, the optimal behavior of Fast would be to mimic the behavior of Slow. For FES strategies with $s \leq 2$, it also proves to be most natural to allow Slow to move on the boundary, but to ignore it when Slow finds the exit first. For

this reason we use FES and SES strategies in this sense. Alternatively, one could also let the non-exploring robot move ε-close to the boundary.

We do not consider SES strategies in our analysis. An optimal SES strategy is to have Slow go to the boundary and explore it (counter) clockwise. Fast follows Slow up to an infinitesimally small distance $\varepsilon > 0$ always staying within the disk interior. The worst case time is $\lim_{\varepsilon \to 0}(1 + 2\pi + \varepsilon) = 1 + 2\pi$.

2.3 Upper Bounds

As a warm-up, consider a "mimic" strategy as a first fast-explores strategy. Both robots set out from the disk center toward the boundary on the same direction each with maximum speed. When Fast reaches the boundary, Slow stops as well. From now on, as Fast explores the boundary in (counter) clockwise fashion, Slow "mimics" Fast's movement by moving on the boundary of a smaller disk with the same center, but with radius $1/s$ instead of 1. When Fast discovers the exit, Slow lies on the corresponding point on the smaller disk and takes a direct line segment of length $1 - 1/s = (s-1)/s$ to it, i.e., Slow traverses the remaining part of a unit disk radius. Overall, in the worst-case, Fast just misses the exit and has to traverse the whole boundary. It takes $1/s$ time for Fast to initially reach the boundary, $2\pi/s$ for Fast to traverse the boundary and another $(s-1)/s$ for Slow to reach the exit. Altogether, we get a $1 + 2\pi/s$ evacuation time.

In the next subsection, we present a more convoluted strategy which outperforms the just described one. Intuitively, the improvement is derived by modifying the behavior of Slow. Instead of mimicking Fast throughout the whole boundary exploration, Slow now tries to be near enough Fast only during the final stages of exploration, that is, when the worst case scenario emerges.

2.3.1 The Half-Chord Strategy for $s \in [2, \infty)$

We now present an FES strategy which we later prove optimal for big enough values of s. The idea for this strategy stems from the proof of the FES lower bound to follow (Lemma 7). When there exists a long chord between two yet unexplored endpoints, an adaptive adversary might place the exit in either endpoint for Fast to discover. In this case, the optimal play for Slow is to be on the midpoint of this chord in order to minimize the time needed till it also reaches the exit.

In the *Half-Chord* strategy, Fast explores the boundary counter clockwise. On the other hand, Slow follows a trajectory with nice properties and reaches the midpoint of a carefully chosen chord whose endpoints capture worst-case evacuation scenarios. Slow reaches the chord midpoint exactly when Fast reaches one of the endpoints (Proposition 2). Then, in the worst-case, Slow has to traverse half the length of this chord to reach the exit (after the exit location is communicated by Fast).

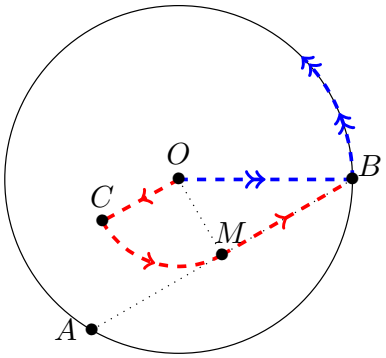


Figure 2.1: The Half-Chord Strategy

The worst-case analysis is performed for $s \in [2, \infty)$. For the strategy details below, please refer to Figure 2.1. The center of the disk is denoted by O. Fast's trajectory is given with *double arrows*, while Slow's is given with *single arrows*. Unless otherwise stated, all angles and arcs are considered in counterclockwise order.

The Strategy. Initially, both robots move in straight lines with an angle $\phi := \angle BOC = \pi + 1/2$ between them until Fast reaches the boundary, that is, for $\frac{1}{s}$ time. Let B be the first boundary point reached by Fast. From now on, Fast's strategy reduces to exploring the boundary. On the other hand, Slow continues on its straight line for another $\frac{1}{s}$ time until it reaches point C, where $|OC| = \frac{2}{s}$. Afterward, for another $\frac{2\arccos(-2/s)-1}{s}$ time, it takes an arc from C to M on the disk with radius $\frac{2}{s}$ centered at O (from now on referred to as disk $(O, \frac{2}{s})$). Finally, Slow traverses MB. Note that, in Figure 2.1, A is the point with arc distance $2\arccos\left(-\frac{2}{s}\right)$ from B.

In Algorithm 2.1, respectively Algorithm 2.2, we provide a more structured and formal main evacuation strategy for Fast, respectively Slow. Bear in mind that if at any time Fast locates the exit, it instantly terminates and informs Slow of the exit location. Then, due to Remark 1, Slow's strategy reduces to following a straight line to the exit.

Algorithm 2.1: Half-Chord for Fast robot

1: Fast lies on point O at time $t = 0$
2: **for** $t \in \left[0, \frac{1}{s}\right]$ **do**
3: | Fast traverses line segment OB
4: **end for**
5: Fast lies on point B at time $t = \frac{1}{s}$
6: **for** $t \in \left(\frac{1}{s}, \frac{1+2\pi}{s}\right]$ **do**
7: | Fast traverses the disk boundary counterclockwise
8: **end for**
9: After exploring the whole boundary, Fast reaches B again at time $t = \frac{1+2\pi}{s}$

Proposition 2. *Fast reaches A exactly when Slow reaches M.*

Algorithm 2.2: Half-Chord for Slow robot

Slow lies on point O at time $t = 0$
for $t \in \left[0, \frac{2}{s}\right]$ **do**
  Slow traverses line segment OC // Phase I
end for
Slow lies on point C at time $t = \frac{2}{s}$
for $t \in \left(\frac{2}{s}, \frac{1+2\arccos(-2/s)}{s}\right]$ **do**
  Slow traverses arc $\widehat{CM}$ on disk $\left(O, \frac{2}{s}\right)$ // Phase II
end for
Slow lies on point M at time $t = \frac{1+2\arccos(-2/s)}{s}$
for $t \in \left(\frac{1+2\arccos(-2/s)}{s}, \frac{1+2\pi}{s}\right]$ **do**
  Slow traverses line segment MB // Phase III
end for

Proof. Fast reaches A after $\frac{1+2\arccos(-2/s)}{s}$ time, since it takes $\frac{1}{s}$ time for it to traverse OB and $\frac{2\arccos(-2/s)}{s}$ time to traverse $\widehat{BA}$. Slow reaches C after time $\frac{2}{s}$. Then, by Algorithm 2.2, it traverses $\widehat{CM}$ for another $\frac{1}{s}(2\arccos(-2/s) - 1)$ time for a total of $\frac{1+2\arccos(-2/s)}{s}$. □

Proposition 3. *M is the midpoint of chord AB.*

Proof. By the strategy, we get $|\widehat{CM}| = \frac{1}{s}(2\arccos(-2/s) - 1)$. Since we work on disk $(O, \frac{2}{s})$, the corresponding angle is $\measuredangle COM = \frac{s}{2}|\widehat{CM}| = \arccos(-2/s) - 1/2$. Let us now consider a parametric representation of the two disks $(O, 1)$ and $(O, \frac{2}{s})$. In such a representation, based on our strategy, we get the following coordinates for point C:

$$C = \left(\frac{2}{s}\cos(\pi + 1/2),\ \frac{2}{s}\sin(\pi + 1/2)\right).$$

Given our knowledge of $\measuredangle COM$, we can extract the coordinates for M as:

$$\begin{aligned} M &= \left(\tfrac{2}{s}\cos(\pi + 1/2 + \arccos(-2/s) - 1/2),\ \tfrac{2}{s}\sin(\pi + 1/2 + \arccos(-2/s) - 1/2)\right) \\ &= \left(-\tfrac{2}{s}\cos(\arccos(-2/s)),\ -\tfrac{2}{s}\sin(\arccos(-2/s))\right) \\ &= \left(\tfrac{4}{s^2},\ -\tfrac{2}{s}\sqrt{1 - \tfrac{4}{s^2}}\right) \end{aligned}$$

where $\cos(\pi + x) = -\cos(x)$, $\sin(\pi + x) = -\sin(x)$, and $\sin(\arccos(x)) = \sqrt{1 - x^2}$ for any x. Now, let us consider points A, B. By the parametric representation, we get the

coordinates:

$$A = (\cos(2\arccos(-2/s)), \sin(2\arccos(-2/s))) = \left(\tfrac{8}{s^2} - 1, -\tfrac{4}{s}\sqrt{1-\tfrac{4}{s^2}}\right),$$
$$B = (\cos(0), \sin(0)) = (1, 0)$$

by the facts that $\cos(2\arccos(x)) = 2x^2 - 1$ and $\sin(2\arccos(x)) = 2x\sqrt{1-x^2}$ for any x. Let us now consider the midpoint of chord AB, namely some point $M' = (x_{M'}, y_{M'})$. We get $x_{M'} = (x_A + x_B)/2 = (8/s^2 - 1 + 1)/2 = 4/s^2$ and $y_{M'} = (y_A + y_B)/2 = (-\frac{4}{s}\sqrt{1-\frac{4}{s^2}} + 0)/2 = -\frac{2}{s}\sqrt{1-\frac{4}{s^2}}$. Noticing that $x_M = x_{M'}$ and $y_M = y_{M'}$ completes the proof. □

Proposition 4. *Fast explores the whole boundary before Slow reaches B.*

Proof. Slow reaches M after $\frac{1+2\arccos(-2/s)}{s}$ time and then has to traverse the line segment MB for another $\sqrt{1-\frac{4}{s^2}}$ time since, by Proposition 3, $|MB| = |BA|/2 = 2\sin(\widehat{BA}/2)/2 = \sin(2\arccos(-2/s)/2) = \sqrt{1-\frac{4}{s^2}}$. Meanwhile, after $\frac{1+2\arccos(-2/s)}{s}$ time, Fast lies on A and then has to traverse $\widehat{AB}$ for another $\frac{2\pi - 2\arccos(-2/s)}{s}$. It's adequate to see that $\sqrt{1-\frac{4}{s^2}} \geq \frac{2\pi - 2\arccos(-2/s)}{s}$ for any $s \geq 2$. □

The aforementioned proposition, together with the fact that it takes $\frac{1+2\pi}{s}$ time for Fast to explore the whole boundary, provides us with the end time for Phase III (see Algorithm 2.2) and the strategy in general.

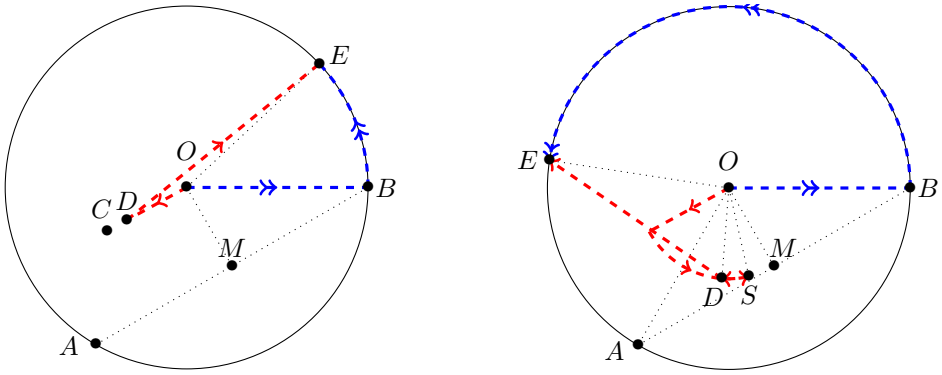
The main result of this section follows by the combination of the upper bounds later proved for Phases I (Lemma 1), II (Lemma 2), and III (Lemma 3) given in Algorithm 2.2.

Theorem 1. *For any $s \geq 2$, the worst-case evacuation time of the Half-Chord strategy is at most $\frac{1+2\arccos\left(-\frac{2}{s}\right)}{s} + \sqrt{1-\frac{4}{s^2}}$.*

Phase I

Lemma 1. *The Half-Chord evacuation strategy takes at most $\frac{(1+2\arccos(-2/s))}{s} + \sqrt{1-\frac{4}{s^2}}$ evacuation time, if the exit is found during Phase I.*

Proof. We need only care about the time $t \in [1/s, 2/s]$, since for less time Fast has not yet reached the boundary. Imagine that the exit is discovered after $(1+a)/s$ time (for $a \in [0,1]$). For a visualization, the reader can refer to Figure 2.2a. Slow has covered $(1+a)/s$ distance on the OC segment, while Fast has explored an a part of $\widehat{BA}$. Slow now takes a segment from its current position (namely D) to the exit E. To compute $|DE|$ we use the law of cosines in $\triangle DOE$. Let $\omega = \measuredangle DOE$. Also, by definition, $a = \widehat{BE} = \measuredangle BOE$ since this is a unit disk and as presented in the strategy description $\measuredangle BOD = \measuredangle BOC = \phi = \pi + 1/2$. Therefore, it follows $\omega = \phi - a = \pi + 1/2 - a$, and further on, it holds



(a) Exit during Phase I ($a = 0.75$) (b) Exit during Phase II ($\tau = 0.3$)

Figure 2.2: Exit during Phases I & II

$\cos(\omega) = \cos(\pi + 1/2 - a) = -\cos(1/2 - a)$. We now compute,

$$|DE| = \sqrt{|OE|^2 + |OD|^2 - 2|OE||OD|\cos(\omega)} = \sqrt{1 + \frac{(1+a)^2}{s^2} + 2\frac{1+a}{s}\cos(1/2 - a)}.$$

Overall, the worst-case evacuation time is given by

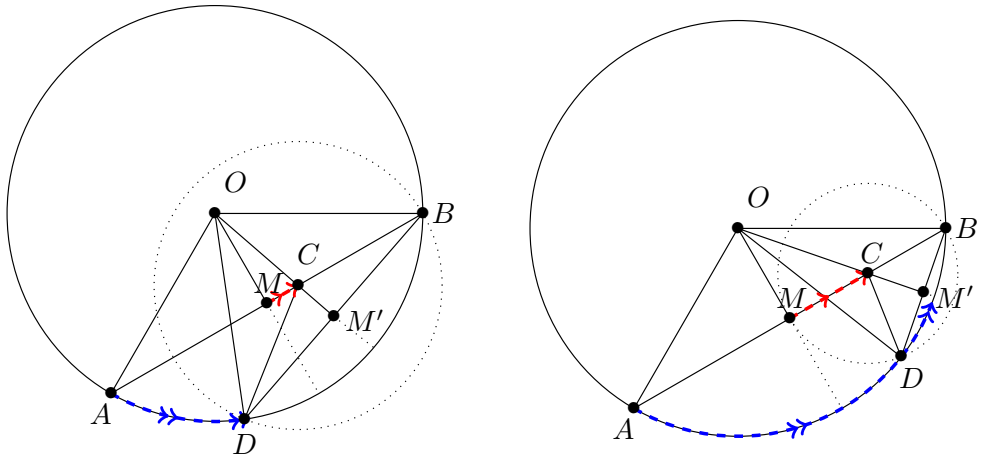
$$\max_{a \in [0,1]} \left\{ \frac{1+a}{s} + \sqrt{1 + \frac{(1+a)^2}{s^2} + 2\frac{1+a}{s}\cos(1/2 - a)} \right\}.$$

To conclude the proof, it suffices to observe that $\frac{2}{s} + \sqrt{1 + \frac{2^2}{s^2} + 2\frac{2}{s}} = \frac{s+4}{s}$ is an upper bound to the above quantity, since $a \leq 1$ and $\cos(\cdot) \leq 1$. Finally, one can verify $\frac{s+4}{s} \leq \frac{1+2\arccos\left(-\frac{2}{s}\right)}{s} + \sqrt{1 - \frac{4}{s^2}}$ for any $s \geq 2$. Let $f(s) = \frac{1+2\arccos\left(-\frac{2}{s}\right)}{s} + \sqrt{1 - \frac{4}{s^2}} - \frac{s+4}{s} = \frac{2\arccos\left(-\frac{2}{s}\right) + \sqrt{s^2-4} - s - 3}{s}$. In the interval $[2, \infty)$, we get: $f(2) = \pi - \frac{5}{2} > 0$, for all $s \geq 2$, $\frac{d}{ds} f(s) = \frac{3 - 2\arccos(-2/s)}{s^2} < 0$ and $\lim_{s \to \infty} f(s) = 0$. □

Phase II

Lemma 2. *The Half-Chord evacuation strategy takes at most* $\frac{1+2\arccos\left(-\frac{2}{s}\right)}{s} + \sqrt{1 - \frac{4}{s^2}}$ *evacuation time, if the exit is found during Phase II.*

Proof. We prove that the worst-case placement for the exit is point A. For the details below, refer to Figure 2.2b. Suppose the exit E is found at the time when Slow lies on point S and has not yet covered a τ part of $\widehat{CM}$. The corresponding central angle is $\frac{s\tau}{2}$, since $\widehat{CM}$ is an arc on $(O, \frac{2}{s})$. At the same time, Fast has not yet explored an $s\tau$ part of $\widehat{BA}$ with a corresponding central angle of size $s\tau$. Then, Slow can move backwards on the boundary of $(O, \frac{2}{s})$ for another τ distance to point D. Now, the central angle from D to M is $\frac{s\tau}{2} + \frac{s\tau}{2} = s\tau$ and matches the central angle between E and A. Thence, due to shifting by the same central angle, we get $\measuredangle EOD = \measuredangle EOA + \measuredangle AOD = \measuredangle DOM + \measuredangle AOD = \measuredangle AOM$.



(a) First case ($\tau = \frac{1}{5}$)

(b) Second case ($\tau = \frac{1}{2}$)

Figure 2.3: Exit during Phase III (example for $s = 4$; exit E lies at the end of Fast's arrow)

Moreover, since $|OD| = |OM| = \frac{2}{s}$ and $|OE| = |OA| = 1$, triangles $\triangle EOD$ and $\triangle AOM$ are congruent meaning that $|ED| = |AM|$. To sum up, if the exit is discovered τ time before Slow reaches M, it takes at most another $\tau + \sqrt{1 - \frac{4}{s^2}}$ time for it to reach it. At the same time, it would take $\tau + \sqrt{1 - \frac{4}{s^2}}$ for it to reach A. Hence, exiting through A is the worst-case scenario and yields a total time of $\frac{1+2\arccos\left(-\frac{2}{s}\right)}{s} + \sqrt{1 - \frac{4}{s^2}}$. □

Phase III

Lemma 3. *The Half-Chord evacuation strategy takes at most* $\frac{1+2\arccos\left(-\frac{2}{s}\right)}{s} + \sqrt{1 - \frac{4}{s^2}}$ *evacuation time, if the exit is found during Phase III.*

Proof. Since $\frac{1+2\arccos\left(-\frac{2}{s}\right)}{s}$ time has passed at the beginning of Phase III, it suffices to show that at most $\sqrt{1 - \frac{4}{s^2}}$ time goes by when the exit is discovered within $\widehat{AB}$.

Suppose that the exit is discovered τ time units after the beginning of Phase III. Then, Slow lies at C (Figure 2.3), τ distance away from M on the MB segment. On the other hand, Fast lies on E, an $s\tau$ distance away from A on $\widehat{AB}$.

Consider a disk with center C and radius $r = \sqrt{1 - \frac{4}{s^2}} - \tau$. One can notice that (C, r) intersects $(O, 1)$ at two points: one of them is B and the other one is D, where D is included in $\widehat{AB}$, since $|AC| \geq r$ for any choice of $\tau \geq 0$. Moreover, we draw the chord DB and its middle point, say M'. Now, notice that OM' is perpendicular to DB, since DB is a chord of $(O, 1)$ and also that OM' passes through C, since DB is also a chord of (C, r). To conclude, we exhibit that E is included in $\widehat{DB}$. Equivalently, that $|\widehat{AE}| \geq |\widehat{AD}|$. We look into two cases.

First (Figure 2.3a), that $\measuredangle AOD \leq \measuredangle AOM$. In this case, we compute

$$\begin{aligned}\measuredangle AOD &= \measuredangle AOM - \measuredangle DOM \\ &= \measuredangle MOB - \measuredangle DOM \\ &= \measuredangle MOM' + \measuredangle M'OB - \measuredangle DOM \\ &= \measuredangle MOM' + \measuredangle DOM' - \measuredangle DOM \\ &= 2 \cdot \measuredangle MOM'\end{aligned}$$

since $\measuredangle AOM = \measuredangle MOB$ and $\measuredangle M'OB = \measuredangle DOM'$ from the fact that OM (OM') bisects AB (DB). Moreover, $\measuredangle DOM' - \measuredangle DOM = \measuredangle MOM'$. We compute $\measuredangle MOM' = \arctan(s\tau/2)$ by the right triangle $\triangle MOC$. Finally, $\measuredangle AOD = 2\arctan(s\tau/2) \leq s\tau = \measuredangle AOE$, since $\arctan(x) \leq x$ for $x \geq 0$.

For the second case (Figure 2.3b), let $\measuredangle AOD > \measuredangle AOM$. Then, $\measuredangle AOD = \measuredangle AOM + \measuredangle MOD = \measuredangle MOB + \measuredangle MOD = \measuredangle MOM' + \measuredangle M'OB + \measuredangle MOD = \measuredangle MOM' + \measuredangle DOM' + \measuredangle MOD = 2 \cdot \measuredangle MOM'$, again by using the equalities deriving from bisecting the chords. The rest of the proof follows as before. □

2.3.2 The Half-Chord Strategy for $1 \leq s \leq 2$

We first observe that, for $s = 2$, the name "Half-Chord" is slightly misleading, as the points A, B, and M coincide. The time needed for $s = 2$ is, as shown in Theorem 1, $\frac{1+2\pi}{s}$. Note also that the Half-Chord strategy is a BES strategy for $s = 2$.

For $s < 2$, Slow can simply move even slower, namely with speed $\frac{s}{2}$. Using the same paths as for $s = 2$, this provides the same upper bound of $\frac{1+2\pi}{s}$.

Theorem 2. *For $1 \leq s \leq 2$, the (generalized) Half-Chord strategy leads to a $\frac{1+2\pi}{s}$ evacuation time.*

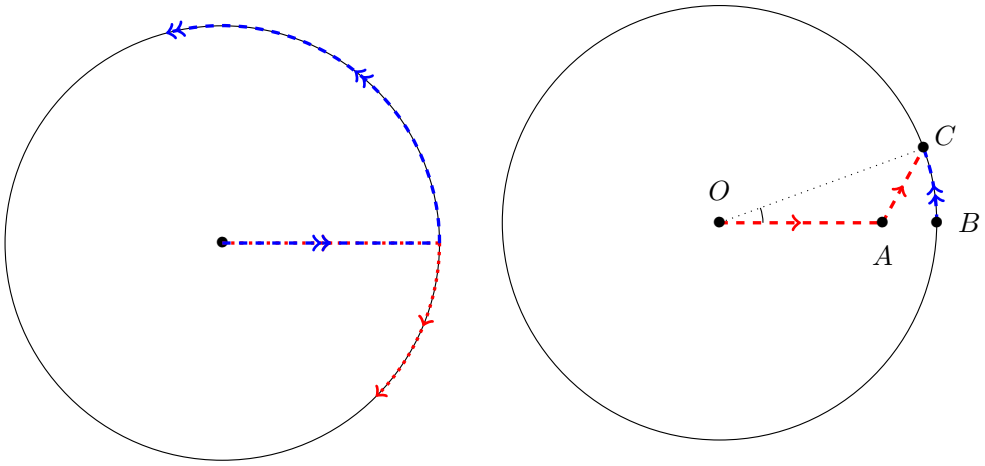
Proof. The two robots follow the exact same trajectories as in the $s = 2$ Half-Chord strategy with a time delay factor of $\frac{2}{s}$. The worst-case evacuation time becomes $\frac{2}{s} \cdot \frac{1+2\pi}{2} = \frac{1+2\pi}{s}$. □

2.3.3 The Both-to-the-Same-Point Strategy

This BES strategy follows the same key idea as the technique presented in [42] where it is proven optimal for the case $s = 1$.

The Strategy

In the *Both-to-the-Same-Point Strategy* (shortly *BSP* strategy), initially both robots set out toward the same boundary point, moving in a straight line. Once they arrive there, they move in opposite directions along the boundary. Without loss of generality, Fast moves counterclockwise along the boundary, while Slow moves clockwise. This goes on, until the exit has been found by either robot or the robots meet each other on the boundary. For



(a) The BSP Strategy

(b) Exit before Slow explores
(Example for $s = 1.8$ and $a = 0.35$)

Figure 2.4: The BSP Strategy and an Evacuation Example

a visualization of the strategy, see Figure 2.4a. Fast's trajectory is given in blue (double arrows), while Slow's in red (single arrows). We restrict the analysis of BSP for $s \in [1, 2]$, since for $s > c_{1.71}$ we later show that this strategy is outperformed.

Exit Before Slow Explores

Lemma 4. *It takes at most* $1 + \sqrt{2 - 2\cos(s-1)}$ *time (where* $s \in [1, 2]$*) for both robots to evacuate in the BSP strategy, when the exit is found before Slow has reached the boundary.*

Proof. Let a stand for the distance Fast has explored on the boundary before finding the exit. Notice that $a \leq s-1 \leq 1$, since a stands for some covered distance before Slow reaches the boundary. The total evacuation time is the time needed for Fast to find the exit and then for Slow to reach it. Let b stand for the latter. Then, the worst-case evacuation time is $\max_{0 \leq a \leq s-1} \left\{ \frac{a+1}{s} + b \right\}$, where $b = \sqrt{1 + \left(\frac{a+1}{s}\right)^2 - 2 \cdot \frac{a+1}{s} \cos(a)}$ by the cosine law in the formed triangle ($\triangle OAC$ in Figure 2.4b with $|OC| = 1$, $|OA| = \frac{1+a}{s}$ and $\measuredangle AOC = a$). Let $f(a, s) = \frac{a+1}{s} + b$. Then,

$$\frac{\partial}{\partial a} f(a, s) = \frac{1}{s} + \frac{\frac{2(a+1)}{s^2} + \frac{2(a+1)\sin(a)}{s} - \frac{2\cos(a)}{s}}{2\sqrt{1 + \left(\frac{a+1}{s}\right)^2 - 2\frac{a+1}{s}\cos(a)}} \geq 0$$

for any $a \leq s - 1$. Consequently, $f(a, s)$ is a non-decreasing function of a in this interval meaning that the maximum is attained on $a = s - 1$. This results to a worst-case time of

$$f(s-1, s) = \frac{s-1+1}{s} + \sqrt{1 + \left(\frac{s-1+1}{s}\right)^2 - 2\frac{s-1+1}{s}\cos(s-1)} = 1 + \sqrt{2 - 2\cos(s-1)}.$$

□

Exit After Slow Explores

Lemma 5. *In the BSP strategy (where $s \in [1,2]$), when the exit is found after Slow has explored some part of the boundary, the evacuation time is at most*

- $\frac{2s+\pi+4}{s+1}$, *when the angle between the two robots is less or equal to π and*
- $1 + 2\sqrt{1 - \frac{1}{(s+1)^2}} + \frac{2\arccos(\frac{1}{-s-1}) - s + 1}{s+1}$ *when the angle is between π and 2π.*

Proof. Let d stand for the distance Slow has covered on the boundary in order to find the exit. Using this notation, the explored part of the boundary is a function of d, s, namely $angle(d,s) = s - 1 + d + s \cdot d = s - 1 + d(s+1)$, since Slow explores distance d, while Fast explores distance $s \cdot d$, and an $s-1$ part has already been covered by Fast till Slow reaches the boundary. The name $angle(\cdot,\cdot)$ is chosen, since the quantity also represents the angle between the robots from the center of the unit disk. We break the analysis into two cases:

- $angle(d,s) \leq \pi$:
  In this case, $s - 1 + d(s+1) \leq \pi$, which results to $d \leq \frac{\pi - s + 1}{s+1}$. Notice that the bound is ≥ 0 for $s \in [1, \pi + 1]$. The worst-case evacuation time is given by computing

$$\max_{0 \leq d \leq \frac{\pi - s + 1}{s+1}} \left\{ 1 + d + 2\sin\left(\frac{d(s+1) + s - 1}{2}\right) \right\}$$

  where the last addend accounts for the chord length needed to be covered by Slow. We denote $g(d,s)$ the function to be maximized. Similarly to before, we see that

$$\frac{\partial}{\partial d} g(d,s) = 1 + (s+1)\cos\left(\frac{(s+1)d + s - 1}{2}\right) \geq 0$$

  for any choice of $s \in [1,2]$ and any $d \in [0, \frac{\pi - s + 1}{s+1}]$, due to the fact that $s + 1 \geq 0$ and $\cos\left(\frac{(s+1)d + s - 1}{2}\right) \geq \cos(\pi/2) = 0$, since $(s+1)d + s - 1 \leq \pi$ and $\cos(\cdot)$ is decreasing in $[0, \pi/2]$. Hence, the maximum is attained at $d = \frac{\pi - s + 1}{s+1}$ for a worst-case time of

$$\begin{aligned} g\left(\tfrac{\pi - s + 1}{s+1}, s\right) &= 1 + \tfrac{\pi - s + 1}{s+1} + 2\sin\left(\frac{(s+1)\frac{\pi - s + 1}{s+1} + s - 1}{2}\right) \\ &= 1 + \tfrac{\pi - s + 1}{s+1} + 2\sin(\pi/2) \\ &= 3 + \tfrac{\pi - s + 1}{s+1} \\ &= \tfrac{2s + \pi + 4}{s+1}. \end{aligned}$$

- $\pi < angle(d,s) < 2\pi$:
  In this case, $d \in (d_{min}, d_{max}) = (\frac{\pi - s + 1}{s+1}, \frac{2\pi - s + 1}{s+1})$. The function to be maximized is again $g(d,s)$. The family of roots for $\frac{\partial g(d,s)}{\partial d} = 0$ is $d = \frac{4\pi n \pm 2 \cdot \arccos(-1/(s+1)) - s + 1}{s+1}$. A

local maximum is attained for

$$d' = \frac{2 \cdot \arccos(-1/(s+1)) - s + 1}{s+1}$$

since d' is the only root lying within (d_{min}, d_{max}) and

$$\begin{aligned} \frac{\partial^2 g(d,s)}{\partial d} \mid_{d=d'} \quad & -\tfrac{1}{2}(s+1)^2 \sin\left(\frac{s+(s+1)\frac{2\arccos\left(-\frac{1}{s+1}\right)-s+1}{s+1}-1}{2}\right) \\ & = -\tfrac{1}{2}(s+1)^2 \sin\left(\arccos\left(-\tfrac{1}{s+1}\right)\right) \\ & = -\tfrac{1}{2}(s+1)^2 \sqrt{1-\left(-\tfrac{1}{s+1}\right)^2} \\ & < 0 \end{aligned}$$

since $\sin(\arccos(x)) = \sqrt{1-x^2}$ for any x. Finally,

$$\begin{aligned} g(d', s) \quad & = 1 + \tfrac{2\cdot\arccos(-1/(s+1))-s+1}{s+1} + 2\sin\left(\frac{s+(s+1)\frac{2\arccos\left(-\frac{1}{s+1}\right)-s+1}{s+1}-1}{2}\right) \\ & = 1 + \tfrac{2\cdot\arccos(-1/(s+1))-s+1}{s+1} + 2\sin\left(\arccos\left(-\tfrac{1}{s+1}\right)\right) \\ & = 1 + \tfrac{2\cdot\arccos(-1/(s+1))-s+1}{s+1} + 2\sqrt{1-\left(-\tfrac{1}{s+1}\right)^2} \end{aligned}$$

is a globally optimal value, since $g(d', s) > g(d_{min}, s) = \frac{2s+\pi+4}{s+1}$ and $g(d', s) > g(d_{max}, s) = \frac{2\pi+2}{s+1}$ for any $s \in [1, 2]$.

Finally, we need not care about the case where Slow finds the exit, since the time taken for Fast to traverse the same chord will be less than the worst-case scenario examined. □

Comparison

For any $s \in [1, 2]$, the maximum (worst-case) upper bound comes from the second case of Lemma 5 and yields the result in Theorem 3.

Theorem 3. *For any $s \in [1, 2]$, BSP requires evacuation time at most*

$$1 + 2\sqrt{1 - \frac{1}{(s+1)^2}} + \frac{2\arccos(-\frac{1}{s+1}) - s + 1}{s+1}.$$

2.3.4 The Fast-Chord Strategy

In the Half-Chord strategy for $s = 2$, we observe that the final point reached after Phase I, i.e., point C, lies on the disk boundary. Thence, after that, Slow explores $\widehat{CB}$, but so does Fast, since by its strategy it explores the whole boundary. This seems like an unnecessary double exploration of this part of the boundary. Thus, we propose a new strategy, where

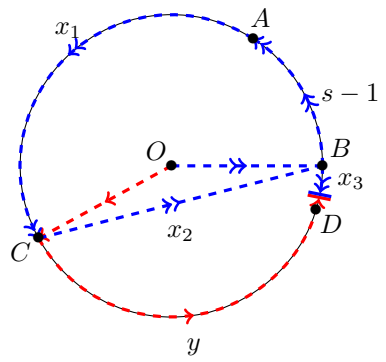


Figure 2.5: The Fast-Chord Family of Strategies

Fast reaches C as usual, but then traverses the CB chord, instead of $\widehat{CB}$. Furthermore, we could vary the position of C, in order for Fast to reach B (for the second time) exactly when Slow reaches D (a point before B) and so get Fast to explore some part of the boundary in clockwise fashion as well. In this case, Slow does not traverse the whole $\widehat{CB}$. Let us now describe more formally this *Fast-Chord* family of strategies. All arcs are considered in *counterclockwise* fashion unless otherwise stated. Below, let $|\widehat{BA}| = s-1$, $x_1 = |\widehat{AC}|$, $x_2 = |CB|$, $x_3 = |\widehat{DB}|$ and $y = |\widehat{CB}|$; see Figure 2.5.

In Algorithms 2.3, 2.4, we define the trajectories followed by Fast and Slow for the Fast-Chord strategy.

The following system of equations describes the relationship between the variable distances:

$$\begin{cases} x_1 + y + x_3 + s - 1 &= 2\pi & \text{(I)} \\ x_2 &= 2\sin\left(\frac{x_3+y}{2}\right) & \text{(II)} \\ x_1 + x_2 &= s \cdot y & \text{(III)} \end{cases}$$

Equation (I) suggests how the disk boundary is partitioned. Equation (II) suggests that x_2 is the chord of an arc with length $x_3 + y$. Equation (III) suggests that Fast traverses x_1 and x_2 at the same time as slow traverses y. That is, since Fast lies on A exactly when Slow lies on C, then Fast arrives at B (for the second time) exactly when Slow arrives at D. The latter happens at time $1 + y = 1 + \frac{x_1+x_2}{s}$. The remaining x_3 part of the boundary can be explored in time $\frac{x_3}{s+1}$, since both robots explore it concurrently until they meet. Hence, within $\frac{x_3}{s+1}$ time, they can explore a distance equal to $s \cdot \frac{x_3}{s+1} + \frac{x_3}{s+1} = (s+1) \cdot \frac{x_3}{s+1} = x_3$. All variables are non-negative representing distance.

The idea behind this paradigm is to try different values for x_3 and then solve the above system to extract x_1, x_2 and y. Nonetheless, due to the $\sin(\cdot)$ function in equation (II), we could not obtain a symbolic solution. Thence, we hereby provide bounds computed *numerically*[1]. For any value of s, we iterate over all possible x_3 values and then solve the

[1]The related source code is available at `https://github.com/yiannislamprou/FastDiskEvacuation`

Algorithm 2.3: Fast-Chord for Fast robot

Fast lies on point O at time $t = 0$
for $t \in \left[0, \frac{1}{s}\right]$ **do**
  Fast traverses line segment OB
end for
Fast lies on point B at time $t = \frac{1}{s}$
for $t \in \left(\frac{1}{s}, 1\right]$ **do**
  Fast traverses arc $\widehat{BA}$ // Phase I
end for
Fast lies on point A at time $t = 1$
for $t \in (1, 1 + \frac{x_1}{s}]$ **do**
  Fast traverses arc $\widehat{AC}$ // Phase IIa
end for
Fast lies on point C at time $t = 1 + \frac{x_1}{s}$
for $t \in \left(1 + \frac{x_1}{s}, 1 + \frac{x_1+x_2}{s}\right]$ **do**
  Fast traverses line segment CB // Phase IIb
end for
Fast lies on point B at time $t = 1 + \frac{x_1+x_2}{s}$
for $t \in \left(1 + \frac{x_1+x_2}{s}, 1 + \frac{x_1+x_2}{s} + \frac{x_3}{s+1}\right]$ **do**
  Fast traverses (clockwise) arc $\widehat{BD}$ till it meets Slow // Phase IIc
end for

above system numerically. For each x_3 value and for each exploration phase, we use a small time step and compute the worst-case evacuation time. We select the x_3 value that minimizes this worst-case time. All this numerical work is implemented in Matlab. We iterate over x_3 in the interval $[0, 2\pi - s + 1]$. The upper bound for x_3 stems from the case $x_1 = y = 0$. Indeed, notice that, for $s = 1$, Fast-Chord is exactly BSP when we set $x_1 = y = 0$. For the time parameter, namely t, we iterate in the interval $\left[0, 1 + \frac{x_1+x_2}{s} + \frac{x_3}{s+1}\right]$. Finally, we use a parametric representation of the disk, where the center O lies on coordinates $(0, 0)$, to calculate the distance between the two robots.

By studying the numerical bounds we obtain via the Fast-Chord method, we state the following result, in comparison to the other two strategies studied in this paper.

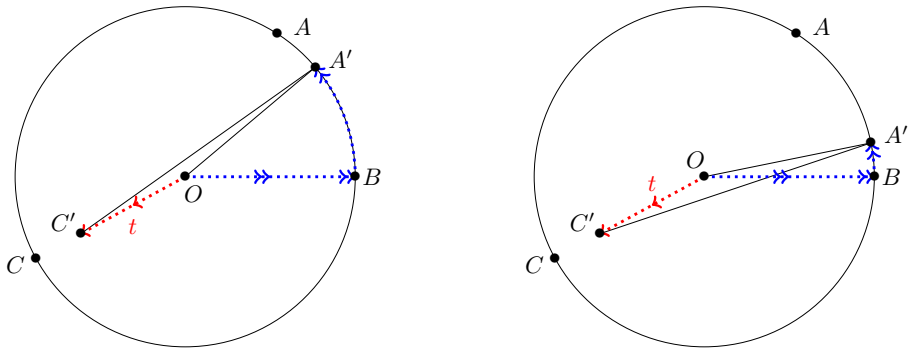
Theorem 4. *Fast-Chord outperforms (Generalized) Half-Chord for $s \in (c_{1.71}, c_{2.07})$. It also outperforms Both-to-the-Same-Point for $s \geq c_{1.71}$.*

We hereby provide the details of the parametric distance calculations we use to validate (up to a certain extent of numerical accuracy) the result in Theorem 4. Below, let $Fast_x$ and $Fast_y$ stand for the (x, y) coordinates of Fast's position and similarly $Slow_x$ and $Slow_y$ for Slow. The distances between the two robots at any given time are as follows (using the phases given in Algorithm 2.3):

Phase I. At time $t \in \left(\frac{1}{s}, 1\right]$, Fast has covered an $st - 1$ part of $\widehat{BA}$ (until point A'), while

Algorithm 2.4: Fast-Chord for Slow robot

Slow lies on point O at time $t = 0$
for $t \in [0, 1]$ **do**
  Slow traverses line segment OC
end for
Slow lies on point C at time $t = 1$
for $t \in (1, 1 + y]$ **do**
  Slow traverses arc $\widehat{CD}$
end for
Slow lies on point D at time $t = 1 + y$
for $t \in \left(1 + y, 1 + y + \frac{x_3}{s+1}\right]$ **do**
  Slow traverses arc $\widehat{DB}$ till it meets Fast
end for



(a) Case (i) (b) Case (ii)

Figure 2.6: Fast-Chord: Exit During Phase I

Slow has covered a t part of OC (until point C'); see Figure 2.6. Their distance is given by applying the cosine law in $\triangle A'OC'$. We compute the *in-triangle* angle $\measuredangle A'OC'$. In case that $\widehat{A'C'} \leq \pi$ (case i), then $\measuredangle A'OC' = \widehat{BC} - \widehat{BA'} = s - 1 + x_1 - (st - 1) = s(1 - t) + x_1$. Otherwise, if $\widehat{A'C'} > \pi$ (case ii), then $\measuredangle A'OC' = 2\pi - \widehat{A'A} - \widehat{AC} = 2\pi - (s - 1 - (st - 1)) - x_1 = 2\pi - s(1 - t) - x_1$. In either case, $|A'C'| = \sqrt{|OA'|^2 + |OC'|^2 - 2|OA'||OC'|\cos(\measuredangle A'OC')} = \sqrt{1 + t^2 - 2t\cos(s(1 - t) + x_1)}$, since $\cos(2\pi - x) = \cos(x)$ for any x.

Phase IIa. At time $t \in \left(1, 1 + \frac{x_1}{s}\right]$, both robots are traversing their respective arcs in counterclockwise fashion. Their positions are the following:

$$(Fast_x, Fast_y) = \left(\cos\left(s\left(t - \frac{1}{s}\right)\right),\ \sin\left(s\left(t - \frac{1}{s}\right)\right)\right),$$

$$(Slow_x, Slow_y) = (\cos(s - 1 + x_1 + t - 1),\ \sin(s - 1 + x_1 + t - 1))$$

where we take into account the initial timestep when they begin traversing their corresponding arcs and the starting position of Slow's arc. Their distance is calculated in the Euclidean norm with the formula $\sqrt{(Fast_x - Slow_x)^2 + (Fast_y - Slow_y)^2}$.

Phase IIb. While Slow continues on the same arc and so its coordinates remain the same as in Phase IIa, Fast is now traversing the CB chord. Its corresponding position is

$$\left(x_C + s\frac{t-1-\frac{x_1}{s}}{x_2}(x_B - x_C),\ y_C + s\frac{t-1-\frac{x_1}{s}}{x_2}(y_B - y_C)\right)$$

where we take into account the direction from C to B, the starting point C, the speed and the initial time step. The normalization factor x_2 provides us with an actual distance instead of a percentage. The above results to $(Fast_x, Fast_y)$ being

$$\left(\cos(s-1+x_1) + s\frac{t-1-\frac{x_1}{s}}{x_2}(1-\cos(s-1+x_1)),\ \sin(s-1+x_1) + s\frac{t-1-\frac{x_1}{s}}{x_2}(-\sin(s-1+x_1))\right)$$

Phase IIc. Again, Slow is always on the same motion and its corresponding parametric equations do not need to change. Fast, on the other hand, commences a clockwise traversal on $\widehat{BD}$ from position 2π with speed s after time step $1 + \frac{x_1+x_2}{s}$.

$$(Fast_x, Fast_y) = \left(\cos\left(2\pi - s\left(t - 1 - \frac{x_1+x_2}{s}\right)\right),\ \sin\left(2\pi - s\left(t - 1 - \frac{x_1+x_2}{s}\right)\right)\right)$$

2.4 Lower Bounds

The main tool behind our lower bounds is the following lemma from [42].

Lemma 6 (Lemma 5 [42])**.** *Consider a boundary of a disk whose subset of total length $u + \epsilon > 0$ has not been explored for some $\epsilon > 0$ and $\pi \geq u > 0$. Then there exist two unexplored boundary points between which the distance along the boundary is at least u.*

2.4.1 Fast Explores

Lemma 7. *Any FES-strategy takes at least*

- $\frac{1+2\pi}{s}$ *time for any* $s \in [1, 2]$ *and*
- $\frac{1+2\arccos\left(-\frac{2}{s}\right)}{s} + \sqrt{1-\frac{4}{s^2}}$ *time for any* $s \geq 2$.

Proof. For any s, Fast needs at least $\frac{1+2\pi}{s}$ time to explore the whole boundary. We now show a better bound for $s \geq 2$. At time $\frac{1+a}{s}$ (where $a \geq 0$), Fast has explored at most an a

part of the boundary. Then, if we consider the time $\frac{1+a-\epsilon}{s}$ (where $\epsilon > 0$), a $2\pi - (a - \epsilon) = 2\pi - a + \epsilon$ subset of the boundary has not yet been explored. We bound $a \in [\pi, 2\pi)$ such that $0 < 2\pi - a \leq \pi$ holds. We now apply Lemma 6 with $u = 2\pi - a$ and ϵ. Thence, there exist two unexplored boundary points between which the distance along the boundary is at least u. Let us now consider the perpendicular bisector of the chord connecting these two points. Depending on which side of the bisector Slow lies, an adversary may place the exit on the boundary point lying at the opposite side. The best case for Slow is to lie exactly on the point of the bisection. That is, Slow will have to cover a distance of at least $\frac{2\sin\left(\frac{u}{2}\right)}{2} = \sin\left(\frac{a}{2}\right)$, where $2\sin\left(\frac{u}{2}\right)$ is the chord length. In this case, the overall evacuation time is equal to $\frac{1+a}{s} + \sin\left(\frac{a}{2}\right)$ and for the best lower bound we compute

$$\max_{\pi \leq a < 2\pi} \left\{ \frac{1+a}{s} + \sin\left(\frac{a}{2}\right) \right\}.$$

The rest of the proof reduces to computing the maximum of this function, namely $f(s, a) = \frac{1+a}{s} + \sin(a/2)$, with respect to a. The first partial derivative is equal to

$$\frac{\partial f(s,a)}{\partial a} = \frac{1}{s} + \frac{1}{2}\cos\left(\frac{a}{2}\right)$$

and the family of solutions to $\frac{\partial f(s,a)}{\partial a} = 0$ is of the form:

$$\left\{ 4\pi n \pm 2\arccos\left(-\frac{2}{s}\right) : n \in \mathbb{Z} \right\}.$$

The only solution which is included in the interval $[\pi, 2\pi)$ is

$$a' = 2\arccos\left(-\frac{2}{s}\right)$$

and it is defined *only for* $s \geq 2$. Moreover, a' is a local maximum, since

$$\frac{\partial^2 f(s,a)}{\partial a} \,|_{a=a'} = -\frac{1}{4}\sin\left(\frac{2\arccos\left(-\frac{2}{s}\right)}{2}\right) = -\frac{1}{4}\sqrt{1 - \frac{4}{s^2}} < 0$$

for any $s \geq 2$. It then suffices to compare $f(s, a')$ to $f(s, \pi) = 1 + \frac{1+\pi}{s}$ and $f(s, 2\pi) = \frac{1+2\pi}{s}$ to prove global optimality. The lower bound is

$$f(s, a') = \frac{1 + 2\arccos\left(-\frac{2}{s}\right)}{s} + \sin\left(\frac{2\arccos\left(-\frac{2}{s}\right)}{2}\right) = \frac{1 + 2\arccos\left(-\frac{2}{s}\right)}{s} + \sqrt{1 - \frac{4}{s^2}}.$$

Finally, notice that the latter bound is equal to $\frac{1+2\pi}{s}$ for $s = 2$ and greater than $\frac{1+2\pi}{s}$ for $s > 2$. □

2.4.2 Both Explore

The following lower bound is a result of applying Lemma 6 to obtain a generalization of the lower bound proved in [42]. The proof considers a timestep when both robots have explored some part of the boundary and lie on the opposite ends of a long chord. Then, an adversary acts according to his best interests. He either places the exit on the end opposite Fast or in the end being farthest to Slow; the latter leading to a chord bisection argument similar to the one used in Lemma 7.

Lemma 8. *Any BES-strategy takes at least*

- $1+\frac{2}{s}\sqrt{1-\frac{s^2}{(s+1)^2}}+\frac{-s+2\arccos\left(-\frac{s}{s+1}\right)+1}{s+1}$ *time for* $s\in[1,2)$,
- $1+\sqrt{1-\frac{4}{(s+1)^2}}+\frac{-s+2\arccos\left(-\frac{2}{s+1}\right)+1}{s+1}$ *for* $s\in[2,c_{4.84}]$ *(where* $c_{4.84}\approx 4.8406$*) and*
- $1+\sin\left(\frac{s-1}{2}\right)$ *time for* $s\in(c_{4.84},2\pi+1)$.

Proof. At time 1, Fast has explored at most $s-1$ distance on the boundary, since it needs $\frac{1}{s}$ time to reach the boundary and in the remaining $\frac{s-1}{s}$ time it can traverse $s\frac{s-1}{s}=s-1$ distance. At time $1+y$, where $y\geq 0$ is a variable, Fast has explored at most an $s-1+sy$ part of the boundary and Slow has explored at most a y part of the boundary. We derive an upper bound for the variable y by noticing that the whole explored part can be strictly less than 2π (otherwise the exit has already been found): $s-1+(s+1)y<2\pi\Rightarrow y<\frac{2\pi-s+1}{s+1}$. Then, the unexplored part is strictly greater than $2\pi-s+1-(s+1)y$. Notice that we need $s<2\pi+1$, otherwise we get $y<0$ which contradicts the $y\geq 0$ initial statement. We let $u=2\pi-s+1-(s+1)y$, where u is the quantity from Lemma 6. We apply the restriction that $u=2\pi-s+1-(s+1)y\leq\pi$, which holds for $y\geq\frac{\pi-s+1}{s+1}$. Moreover, $u=2\pi-s+1-(s+1)y>0$ holds for any $s\geq 1$ given that $y<\frac{2\pi-s+1}{s+1}$.

Now, let us apply Lemma 6: There exist two unexplored points with arc distance $\geq 2\pi-s+1-(s+1)y$, which implies that the chord between them has length at least $2\sin\left(\frac{2\pi-s-(s+1)y+1}{2}\right)=2\sin\left(\frac{s+(s+1)y-1}{2}\right)$. An adversary can put the exit on any of the two endpoints. If Slow reaches an endpoint first (case I), then the exit is placed on the other side, such that Slow has to traverse the chord. If Fast reaches an endpoint first, then the exit is placed either on the other side (case II), meaning that Fast has to traverse the chord, or on the endpoint that lies the farthest from Slow's current position (case III), meaning that Slow has to traverse at least half the chord. We assume that both the robots and the adversary behave optimally. Hence, the robots will always avoid case I. Then, the adversary will apply case II, for $s\in[1,2)$, and III for $s\geq 2$. Let $y_{min}=\max\{0,\frac{\pi-s+1}{s+1}\}$ and $y_{max}=\frac{2\pi-s+1}{s+1}$. Totally, the worst-case evacuation time is given by

- $\max_{y\in[y_{min},y_{max})}\left\{1+y+\frac{2}{s}\sin\left(\frac{s+(s+1)y-1}{2}\right)\right\}$, when in case II and
- $\max_{y\in[y_{min},y_{max})}\left\{1+y+\sin\left(\frac{s+(s+1)y-1}{2}\right)\right\}$, when in case III.

The rest of the proof reduces to computing the maximum of these functions, with respect to y.

Case II Let $f_{II}(y,s) = 1 + y + \frac{2}{s}\sin\left(\frac{(s+1)y+s-1}{2}\right)$ be the function arising from case II. We only analyze the function for $s \in [1,2)$, since for $s \geq 2$ it is easy to see that case III provides a stronger lower bound. We compute

$$\frac{\partial}{\partial y}\left(1 + y + \frac{2}{s}\sin\left(\frac{(s+1)y+s-1}{2}\right)\right) = 1 + \frac{(s+1)\cos\left(\frac{(s+1)y+s-1}{2}\right)}{s}$$

which gives the following family of roots

$$y = \frac{4\pi n \pm 2\arccos\left(-\frac{s}{s+1}\right) - s + 1}{s+1} \quad (n \in \mathbb{Z}).$$

The only root (and so potential maximum of the function) that lies within $[y_{min}, y_{max}]$ is

$$y' = \frac{2\arccos\left(-\frac{s}{s+1}\right) - s + 1}{s+1}.$$

Moreover, we can see that $y' > 0$ for $s \in [1,2)$ as needed, since y' represents distance. We demonstrate concavity at y' by computing

$$\begin{aligned}
\tfrac{\partial^2 f_{II}(y,s)}{\partial y}\,|_{y=y'} &= \tfrac{\partial}{\partial y}\left(1 + \frac{(s+1)\cos(\left(\frac{(s+1)y+s-1}{2}\right)}{s}\right)|_{y=y'} \\
&= -\frac{(s+1)^2\sin\left(\frac{(s+1)y+s-1}{2}\right)}{2s}\,|_{y=y'} \\
&= -\frac{(s+1)^2\sqrt{1-\frac{s^2}{(s+1)^2}}}{2s} < 0
\end{aligned}$$

and we compute the value of f_{II} as

$$\begin{aligned}
f_{II}(y',s) &= f_{II}\left(\tfrac{2\arccos\left(-\frac{s}{s+1}\right)-s+1}{s+1}, s\right) \\
&= 1 + \tfrac{2\arccos\left(-\frac{s}{s+1}\right)-s+1}{s+1} + \tfrac{2}{s}\sin\left(\arccos\left(-\tfrac{s}{s+1}\right)\right) \\
&= 1 + \tfrac{2\arccos\left(-\frac{s}{s+1}\right)-s+1}{s+1} + \tfrac{2\sqrt{1-\frac{s^2}{(s+1)^2}}}{s}.
\end{aligned}$$

Finally, we compute the values at the interval endpoints

$$f_{II}(y_{min}, s) = f_{II}\left(\frac{\pi+1-s}{s+1}, s\right) = 1 + \frac{\pi+1-s}{s+1} + \frac{2}{s}\sin(\pi/2) = \frac{\pi s + 4s + 2}{s(s+1)},$$

$$f_{II}(y_{max}, s) = f_{II}\left(\frac{2\pi+1-s}{s+1}, s\right) = 1 + \frac{2\pi+1-s}{s+1} + \frac{2}{s}\sin(\pi) = \frac{2\pi+2}{s+1}.$$

It suffices to verify they are always less to $f_{II}(y', s)$ for $s \in [1, 2]$.

Case III Let $f_{III}(y, s) = 1 + y + \sin\left(\frac{(s+1)y+s-1}{2}\right)$ stand for the function to be maximized arising from Case III. We follow the same steps as before.

$$\frac{\partial f_{III}(y, s)}{\partial y} = 1 + \frac{(s+1)\cos\left(\frac{(s+1)y+s-1}{2}\right)}{2}$$

gives the following family of roots

$$y = \frac{4\pi n \pm 2\arccos\left(\frac{2}{s+1}\right) - s + 1}{s+1} \quad (n \in \mathbb{Z}).$$

The only root (and thus potential maximum of the function) that lies within $[y_{min}, y_{max}]$ is $y' = \frac{2\arccos\left(\frac{2}{s+1}\right)-s+1}{s+1}$. Moreover, we can see that $y' > 0$ holds *only* for $s \in [2, c_{4.84})$, where $c_{4.84} \approx 4.8406$. We demonstrate concavity at y' by

$$\begin{aligned}
\frac{\partial^2 f_{III}(y,s)}{\partial y} \mid_{y=y'} &= \frac{\partial}{\partial y}\left(1 + \frac{(s+1)\cos\left(\frac{(s+1)y+s-1}{2}\right)}{2}\right) \mid_{y=y'} \\
&= -\frac{(s+1)^2 \sin\left(\frac{(s+1)y+s-1}{2}\right)}{4} \mid_{y=y'} \\
&= -\frac{(s+1)^2\sqrt{1-\frac{4}{(s+1)^2}}}{4} < 0
\end{aligned}$$

and compute the value of f_{III} as

$$\begin{aligned}
f_{III}(y', s) &= f_{III}\left(\frac{2\arccos\left(\frac{2}{s+1}\right)-s+1}{s+1}, s\right) \\
&= 1 + \frac{2\arccos\left(\frac{2}{s+1}\right)-s+1}{s+1} + \sin\left(\arccos\left(\frac{2}{s+1}\right)\right) \\
&= 1 + \frac{2\arccos\left(\frac{2}{s+1}\right)-s+1}{s+1} + \sqrt{1-\frac{4}{(s+1)^2}}.
\end{aligned}$$

Finally, we compute $f_{III}(y_{min}, s)$, $f_{III}(y_{max}, s)$ which are less than $f_{III}(y', s)$ for $s \in [2, c_{4.84})$, where $c_{4.84} \approx 4.84$.

$$f_{III}(y_{min}, s) = f_{III}(0, s) = 1 + \sin\left(\frac{s-1}{2}\right)$$

$$f_{III}(y_{max}, s) = f_{III}\left(\frac{2\pi+1-s}{s+1}, s\right) = 1 + \frac{2\pi+1-s}{s+1} + \sin(\pi) = \frac{2\pi+2}{s+1}$$

For the case $s \geq c_{4.84}$, we need only consider the endpoints of the $[y_{min}, y_{max}]$ interval

as potential maxima: $f_{III}(y_{min}) \geq f_{III}(y_{max})$ for $s \in [c_{4.84}, 2\pi + 1)$. □

The above lower bound loses its value as s grows. This happens due to the fact that in the proof we consider only a specific moment of a both-explore strategy, where both robots have already explored some part of the boundary. Hence, there is a need to capture a lower bound for the case where Slow has not explored any part of the boundary yet. This is possible, since we can apply an FES lower bound idea when s is big enough.

Lemma 9. *Any BES-strategy takes at least*

- $1 + \sin\left(\frac{s-1}{2}\right)$ *time for* $s \in (\pi + 1, c_{4.97})$, *where* $c_{4.97} \approx 4.9699$, *and*
- $\frac{1+2\arccos\left(-\frac{2}{s}\right)}{s} + \sqrt{1 - \frac{4}{s^2}}$ *time for* $s \geq c_{4.97}$.

Proof. One need only notice that, for $a = s - 1 > \pi$, at time $\frac{1+a-\epsilon}{s}$, a $2\pi - a + \epsilon$ part of the boundary is yet unexplored, where $2\pi - a \leq \pi$. Moreover, Slow has not reached the boundary yet. Hence, we can view this as a fast-explores subcase. Then, we can compute $\max_{a \in [\pi, \min\{s-1, 2\pi\}]} \left\{\frac{1+a}{s} + \sin\left(\frac{a}{2}\right)\right\}$.

Let $f(a, s) = \frac{1+a}{s} + \sin\left(\frac{a}{2}\right)$ be the emerging function that needs to be maximized for $a \in [\pi, \min\{s - 1, 2\pi\}]$. This function is already analyzed in the proof of Lemma 7. Nevertheless, we now need to reconsider it, since the underlying domain depends on s. For $s \geq 2\pi + 1$, $\min\{s - 1, 2\pi\} = 2\pi$ and so the analysis proceeds as before yielding a lower bound of $\sqrt{1 - \frac{4}{s^2}} + \frac{1+2\arccos(-2/s)}{s}$. Let us now consider $s \in [\pi + 1, 2\pi + 1)$. The selected derivative root is again $a' = 2\arccos\left(-\frac{2}{s}\right)$. Nonetheless, one ought to notice that $2\arccos\left(-\frac{2}{s}\right) \leq s - 1$ *only for* $s \geq c_{4.97}$, where $c_{4.97} \simeq 4.9699$. Now, let us compare $f(a', s)$ to $f(\pi, s)$ and $f(s-1, s)$ (i.e. the values at the endpoints of the interval). We get $f(a', s) = \sqrt{1 - \frac{4}{s^2}} + \frac{1+2\arccos(-2/s)}{s}$ as before, $f(\pi + 1, s) = 1 + \frac{1+\pi}{s}$ and $f(s - 1, s) = 1 + \sin\left(\frac{s-1}{2}\right)$.

One can notice that $f(a', s)$ prevails for $s \geq c_{4.97}$, while $f(s - 1, s)$ is greater to $f(\pi, s)$ for $s \in [c_{4.97}, \infty)$. On the other hand, for $s \in (\pi + 1, c_{4.97})$, $f(s - 1, s) \geq f(\pi, s)$. □

The following lemma encompasses the above BES lower bounds in Lemmata 8 and 9 by taking the maximum for each value of s.

Lemma 10. *Any BES-strategy takes at least*

- $1 + \frac{2}{s}\sqrt{1 - \frac{s^2}{(s+1)^2}} + \frac{-s+2\arccos\left(-\frac{s}{s+1}\right)+1}{s+1}$ *time for* $s \in [1, 2)$,
- $1 + \sqrt{1 - \frac{4}{(s+1)^2}} + \frac{-s+2\arccos\left(-\frac{2}{s+1}\right)+1}{s+1}$ *for* $s \in [2, c_{4.84}]$,
- $1 + \sin\left(\frac{s-1}{2}\right)$ *time for* $s \in (c_{4.84}, c_{4.97})$ *and*
- $\frac{1+2\arccos(-2/s)}{s} + \sqrt{1 - \frac{4}{s^2}}$ *time for* $s \in [c_{4.97}, \infty)$.

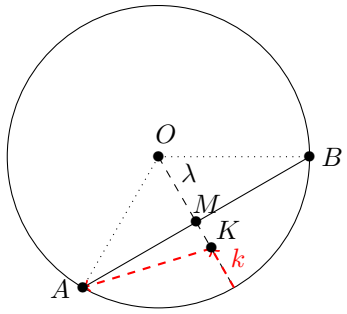


Figure 2.7: An Improved BES Lower Bound

2.4.3 An Improvement for BES

We now obtain numerical values for a stronger BES lower bound by performing a more complex analysis on the *Original* BES lower bound proof given in Lemma 8. The main idea behind the improvement is to provide a better bound for the subcase when the adversary places the exit on the farthest endpoint from Slow's current position. Apparently, the best play for Slow is to lie exactly on the midpoint of the chord with the unexplored endpoints. Nevertheless, in order for Slow to be there, it needs to spend some of its time, originally destined for exploration, within the disk interior. We hereby examine the best possible scenario for Slow in terms of its distance from the midpoint following the above reasoning. Let us refer to this lower bound as *Improved* BES.

Lemma 11. *Improved* BES *is greater or equal to Original* BES *for any* $s \geq 1$.

Proof. At time $1+y$, where $y \geq 0$ is a variable, Fast has explored at most an $s-1+sy$ part of the boundary and Slow has explored at most a y part of the boundary. Now suppose that Slow has spent k time, where $k \in [0, y]$, *not exploring* the boundary, i.e. moving within the disk interior.

Notice that it takes $1 + \frac{2\pi-s+1}{s+1}$ time for the whole perimeter to be explored, when both robots are only exploring after timestep 1. Thence, we upper-bound $y \leq \frac{2\pi-s+1}{s+1}$. To lower-bound y, we restrict the unexplored part $u = 2\pi - s + 1 - (s+1)y + k \leq \pi$. That is, we get $y \geq \max\{\frac{\pi-s+1+k}{s+1}, 0\}$. Moreover, $u > 0$ is already covered by the aforementioned upper bound.

Now, we are ready to apply Lemma 6: There exist two unexplored points (say A, B) with arc distance $\geq 2\pi - s + 1 - (s+1)y + k$, which implies that the chord between them has length at least $2\sin\left(\frac{2\pi-s+1-(s+1)y+k}{2}\right) = 2\sin\left(\frac{s-1+(s+1)y-k}{2}\right)$. An adversary could place the exit on any of the two endpoints. If Slow reaches an endpoint first (case I), then the exit is placed on the other side, such that Slow has to traverse the chord. If Fast reaches an endpoint first, then the exit is placed either on the other side (case II), meaning that Fast has to traverse the chord, or on the endpoint that lies the farthest from Slow's current position (case III), meaning that Slow has to traverse at least half the chord. We assume

that both the robots and the adversary behave optimally. Hence, the robots will always avoid case I.

Let us now examine more carefully what happens in case III. For a depiction of the proof, see Figure 2.7. The ideal location for Slow is to lie exactly on the chord midpoint, say M. Nevertheless, this may not be possible due to it only spending k time within the disk interior. Let us consider the minimum distance from the chord midpoint to the boundary. This is exactly $1-\lambda$, where $\lambda = |OM|$ is the distance from the midpoint to the center of the disk. Notice that OM intesects AB *perpendicularly*, since M is the midpoint of chord AB. Using the Pythagorean theorem in $\triangle AMO$, we get $\lambda = \sqrt{1-\sin^2\left(\frac{s-1+(s+1)y-k}{2}\right)}$. If we consider the case when $1-\lambda > k$, then the ideal position for Slow is to lie k distance away from the boundary and on the extension of OM (i.e. on point K). From there, Slow can take a straight line to the exit, yielding a $\sqrt{\sin^2\left(\frac{s-1+(s+1)y-k}{2}\right)+(1-\lambda-k)^2}$ distance again by the Pythagorean theorem, now in $\triangle AMK$.

To conclude, Slow will try to minimize this straight line distance over k, while the adversary will select a case between II and III that maximizes the total distance. Overall, the optimization problem reduces to computing:

$$\max_{y\in[y_{min},y_{max})}\left\{1+y+\max\left\{\begin{array}{l}\min\limits_{k\in[0,y]}\frac{2}{s}\sin\left(\frac{s-1+(s+1)y-k}{2}\right),\\ \min\limits_{k\in[0,y]}\sqrt{\sin^2\left(\frac{s-1+(s+1)y-k}{2}\right)+\max\{1-\lambda-k,0\}^2}\end{array}\right\}\right\}. \quad (2.1)$$

Note that the above bound matches the original one for $1-\lambda < k$.

Last but not least, we need also consider the case where the adversary chooses to place the exit on the last boundary point to be explored. In the current setting, it takes at least $\frac{u}{s+1} = \frac{2\pi-s+1-(s+1)y+k}{s+1}$ extra time for both robots to explore the rest of the boundary, since Fast explores $s\frac{u}{s+1}$ while Slow explores $\frac{u}{s+1}$ for a total distance of u. Overall, we are looking to compute $\max\limits_{y\in[y_{min},y_{max})}\left\{1+y+\frac{2\pi-s+1-(s+1)y}{s+1}\right\}$, since Slow wishes to minimize k. Due to the inherent complexity of the optimization problem (2.1), we compute *numerical* bounds[2]. The two min expressions are computed and the maximum of them is chosen as the best-play scenario for an adversary. The computational work is done in Matlab, where we iterate over feasible values of y and k with a step of 10^{-3}. For Fast's speed s, we iterate with a step of 10^{-1}. The resulting bounds show that, for all $s \in [1, 2\pi+1)$, this lower bound is greater or equal to the lower bound given in Lemma 8 with $k=0$ *always* selected as the minimizer. □

[2]The related source code is available at `https://github.com/yiannislamprou/FastDiskEvacuation`

2.5 Comparison of Bounds

Regarding the lower bounds, for each value of s we select the minimum (weakest) lower bound between the (maximum) BES and FES ones as our overall lower bound. We see (Fig. 2.8) that Improved BES (Lemma 11) is strictly stronger than Original BES (Lemma 10) for any $s \geq c_{1.71} \approx 1.71$. Moreover, Improved BES is stronger than the FES lower bound (Lemma 7) for $s \geq c_{2.75} \approx 2.75$.

As far as the upper bounds are concerned, we notice (Fig. 2.9) that Half-Chord (Theorem 1) outperforms BSP (Theorem 3) for any $s \geq c_{1.86} \approx 1.856$. Besides, Fast-Chord (Theorem 4) outperforms BSP for any $s \geq c_{1.71} \approx 1.71$. Finally, Fast-Chord outperforms Half-Chord for any $s \leq c_{2.07} \approx 2.072$. That is, the introduction of Fast-Chord yields a better upper bound for any $s \in [c_{1.71}, c_{2.07}]$.

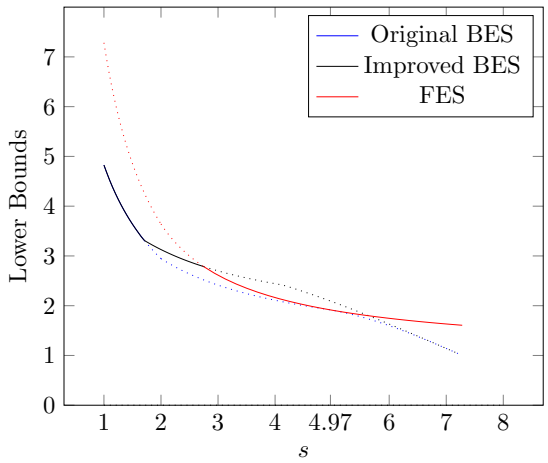


Figure 2.8: Comparison of lower bounds

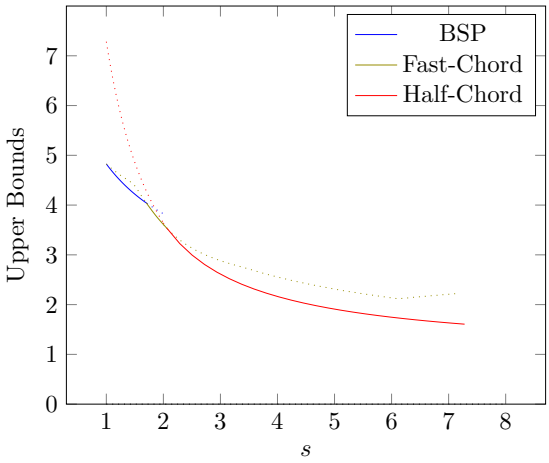


Figure 2.9: Comparison of upper bounds

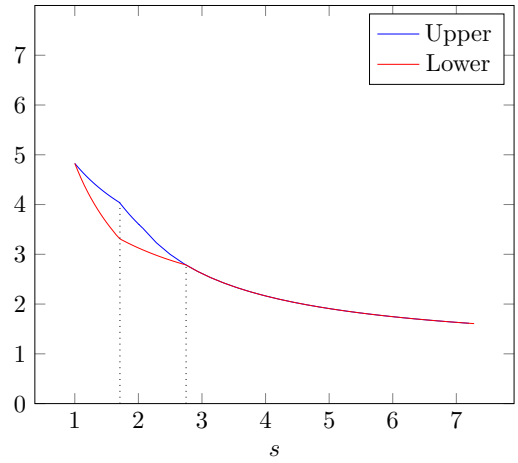


Figure 2.10: Dominant Lower vs Upper Bounds

By comparing upper and lower bounds, we see (Fig. 2.10) that Half-Chord is optimal for $s \geq c_{2.75}$, since the matching FES lower bound is the weakest in this interval. On the other hand, for $s < c_{2.75}$ the ratio between the bounds is at most 1.22 (maximized when $s = c_{1.71}$), where the strategy changes from BSP to Fast-Chord. The best strategy to use is BSP when $s < c_{1.71}$, Fast-Chord when $c_{1.71} < s < c_{2.07}$ and Half-Chord for $s \geq c_{2.07}$.

2.6 Open Problems

Optimality for the case $1 < s < c_{2.75}$ remains open. In this gray area, the main difficulty is understanding when it becomes necessary to make the transition from a BES to an FES strategy. As indicated by our introduction of Fast-Chord, which outperforms BSP and Half-Chord in the interval $(c_{1.71}, c_{2.07})$, the potential strategies might need to get even more convoluted to capture the diminishing speed ratio.

Regarding future work on this topic, one could consider extending these results to a more-than-two-robots evacuation scenario. Moreover, the non-wireless case for two-robots fast evacuation seems to be an even more challenging open problem given that exact optimality is complex to obtain even for $s = 1$ ([23, 47]). Finally, other environments could be examined, e.g. polygonal ones [41], or more realistic robotic settings where the environment becomes more perplexing, e.g., including spatial obstacles or communication restrictions.

Chapter 3

Dynamic Graph Exploration

3.1 Introduction

In the modern era of the Internet, modifications in a network topology can occur extremely frequently and in a disorderly way. Communication links may fail from time to time, while connections amongst terminals may appear or disappear intermittently. Thus, classical (static) network theory fails to capture such ever-changing processes. In an attempt to fill this void, different research communities have given rise to a variety of theories on *dynamic networks*. In the context of algorithms and distributed computing, such networks are usually referred to as *temporal graphs* [96]. A temporal graph is represented by a (possibly infinite) sequence of subgraphs of the same static graph. That is, the graph is *evolving* over a series of (discrete) time steps under a set of deterministic or stochastic rules of evolution. Such a rule can be edge- or graph-specific and may take as input graph instances observed in previous time steps.

In this chapter, we focus on stochastically-evolving temporal graphs. We define a model of evolution, where there exists a single stochastic rule, which is applied *independently* to each edge. Intuitively, such an independence assumption models latency/noise issues referring to each specific network link. Our model is general in the sense that the underlying static graph is allowed to be a general connected graph, i.e., with no further constraints on its topology, and the stochastic rule can include any finite number of past observations.

Assume now that a single mobile agent is placed on an arbitrary vertex of a temporal graph evolving under the aforementioned model. Next, the agent performs a simple random walk; at each time step, after the graph instance is fixed according to the model, the agent chooses uniformly at random a vertex amongst the neighbors of its current vertex and visits it. The *cover time* of such a walk is defined as the expected number of time steps until the agent has visited each vertex at least once. Herein, we prove some first bounds on the cover time for a simple random walk as defined above, mostly via the use of Markovian theory.

Random walks constitute a very important primitive in terms of distributed computing. Examples include their use in information dissemination [3] and random network

structure [9]; also, see the short survey in [26]. In this work, we consider a single random walk as a fundamental building block for other more distributed scenarios to follow.

3.1.1 Related Work

A paper very relevant to our work is the one of Clementi, Macci, Monti, Pasquale and Silvestri [36], where they consider the flooding time in *Edge-Markovian* dynamic graphs. In such graphs, each edge independently follows a Markovian rule and their model appears as a special case of ours (matches our case $k = 1$). Further work under this Edge-Markovian paradigm on flooding and information dissemination includes [11, 38].

Another related work is the one of Avin, Koucký and Lotker [7], who define the notion of a *Markovian Evolving Graph*, i.e., a temporal graph evolving over a set of graphs $G_1, G_2, \ldots$, where the process transits from G_i to G_j with probability p_{ij}, and consider random walk cover times. Note that their approach becomes computationally intractable if applied to our case; each of the possible edges evolves independently, thence causing the state space to be of size 2^m, where m is the number of possible edges in our model.

Clementi, Monti, Pasquale and Silvestri [37] study the broadcast problem, when at each time step the graph is selected according to the well-known $G_{n,p}$ model. Also, Yamauchi, Izumi and Kamei [113] study the rendezvous problem for two agents on a ring, when each edge of the ring independently appears at every time step with some fixed probability p.

Moving to a more general scope, research in temporal networks is of interdisciplinary interest, since they are able to capture a wide variety of systems in physics, biology, social interactions and technology. For a view of the big picture, see the review in [73]. There exist several papers considering, mostly continuous-time, random walks on different models of temporal networks: In [107], they consider a walker navigating randomly on some specific empirical networks. Rocha and Masuda [104] study a lazy version of a random walk, where the walker remains to its current vertex according to some sojourn probability. In [54], they study the behavior of a continuous time random walk on a stationary and ergodic time varying dynamic graph. Lastly, random walks with arbitrary waiting times are studied in [48], while random walks on stochastic temporal networks are surveyed in [72].

In the analysis to follow, we employ several seminal results around the theory of random walks and Markov chains. For random walks, we base our analysis on the seminal work in [3] and the electrical network theory presented in [30, 50]. For results on Markov chains, we cite textbooks [68, 98].

3.1.2 Our Results

We define a general model of stochastically-evolving graphs, where each possible edge evolves independently, but all of them evolve following the same stochastic rule. Furthermore, the stochastic rule may take into account the last k states of a given edge. The motivation for such a model lies in several practical examples from networking where the

existence of an edge in the recent past means it is likely to exist in the near future, e.g., for telephone or Internet links. In some other cases, existence may mean that an edge has "served its purpose" and is now unlikely to appear in the near future, e.g., due to a high maintenance cost. The model is a discrete-time one following previous work in the computer science literature. Moreover, as a first start and for mathematical convenience, it is formalized as a synchronous system, where all possible edges evolve concurrently in distinct rounds (each round corresponding to a discrete time step).

Special cases of our model have appeared in previous literature, e.g., in [37,113] for $k = 0$ and in the line of work starting from [36] for $k = 1$, however they only consider special graph topologies (like ring and clique). On the other hand, the model we define is general in the sense that no assumptions, aside from connectivity, are made on the topology of the underlying graph and any amount of history is allowed into the stochastic rule. Thence, we believe it can be valued as a basis for more general results to follow capturing search or communication tasks in such dynamic graphs.

We hereby provide the first known bounds relative to the cover time of a simple random walk taking place in such stochastically evolving graphs for $k = 0$. To do so, we make use of a simple, yet fairly useful, modified random walk, namely the *Random Walk with a Delay* (*RWD*), where at each time step the agent is choosing uniformly at random from the incident edges of the static underlying graph and then waits for the chosen edge to become alive in order to traverse it. Despite the fact that this strategy may not sound naturally-motivated enough, it can act as a handy tool when studying other, more natural, random walk models as in the case of this paper. Indeed, we study the natural random walk on such graphs, namely the *Random Walk on What's Available* (*RWA*), where at each time step the agent only considers the currently alive incident edges and chooses to traverse one out of them uniformly at random.

For the case $k = 0$, that is, when each edge appears at each round with a fixed probability p regardless of history, we prove that the cover time for *RWD* is upper bounded by C_G/p, where C_G is the cover time of a simple random walk on the (static) underlying graph G. The result can be obtained both by a careful mapping of the *RWD* walk to its corresponding simple random walk on the static graph and by generalizing the standard electrical network theory literature in [30, 50]. Later, we proceed to prove that the cover time for *RWA* is between $C_G/(1-(1-p)^{\Delta})$ and $C_G/(1-(1-p)^{\delta})$ where δ, respectively Δ, is the minimum, respectively maximum, degree of the underlying graph. The main idea here is to reduce *RWA* to an *RWD* walk, where at each step the traversal delay is lower, respectively upper, bounded by $(1-(1-p)^{\delta})$, respectively $(1-(1-p)^{\Delta})$.

For $k = 1$, the stochastic rule takes into account the previous, one time step ago, state of the edge. If an edge was not present, then it becomes alive with probability p, whereas if it was alive, then it dies with probability q. For *RWD*, we show a C_G/ξ_{min} upper bound by considering the minimum probability guarantee of existence at each round, i.e., $\xi_{min} =$

$\min\{p, 1-q\}$. Similarly, we show a C_G/ξ_{max} lower bound, where $\xi_{max} = \max\{p, 1-q\}$.

Consequently, we demonstrate an exact, exponential-time approach to determine the precise cover time value for a general setting of stochastically-evolving graphs, including also the edge-independent model considered in this paper.

3.1.3 Outline

In Section 3.2, we define our model of stochastically-evolving graphs in a more rigorous fashion. Afterwards, in Sections 3.3 and 3.4, we provide the analysis of our cover time bounds when for determining the current state of an edge we take into account its last 0 and 1 states, respectively. In Section 3.5, we demonstrate an exact approach for determining the cover time for general stochastically-evolving graphs. Finally, in Section 3.6, we cite some concluding remarks.

3.2 The Edge-Uniform Evolution Model

Let us define a general model of a dynamically evolving graph. Let $G = (V, E)$ stand for a simple, *connected* graph, from now on referred to as the *underlying graph* of our model. The number of vertices is given by $n = |V|$, while the number of edges is denoted by $m = |E|$. For a vertex $v \in V$, let $N(v) = \{u : (v, u) \in E\}$ stand for the *open neighborhood* of v and $d(v) = |N(v)|$ for the *(static) degree* of v. Note that we make no assumptions regarding the topology of G, besides connectedness. We refer to the edges of G as the *possible edges* of our model. We consider evolution over a sequence of discrete time steps (namely $0, 1, 2, \ldots$) and denote by $\mathcal{G} = (G_0, G_1, G_2, \ldots)$ the infinite sequence of graphs $G_t = (V_t, E_t)$, where $V_t = V$ and $E_t \subseteq E$. That is, G_t is the graph appearing at time step t and each edge $e \in E$ is either *alive* (if $e \in E_t$) or *dead* (if $e \notin E_t$) at time step t.

Let R stand for a *stochastic rule* dictating the probability that a given possible edge is alive at any time step. We apply R at each time step and at each edge *independently* to determine the set of currently alive edges, i.e., the rule is *uniform* with regard to the edges. In other words, let e_t stand for a random variable where $e_t = 1$, if e is alive at time step t, or $e_t = 0$, otherwise. Then, R determines the value of $\Pr(e_t = 1|H_t)$ where H_t is also determined by R and denotes the history length, i.e., the values of $e_{t-1}, e_{t-2}, \ldots$, considered when deciding for the existence of an edge at time step t. For instance, $H_t = \emptyset$ means no history is taken into account, while $H_t = \{e_{t-1}\}$ means the previous state of e is taken into account when deciding for its current state.

Overall, the aforementioned *Edge-Uniform Evolution* model (shortly *EUE*) is defined by the parameters G, R and some initial input instance G_0. In the following sections, we consider some special cases for R and provide some first bounds for the cover time of G under this model. Each time step of evolution consists of two stages: in the first stage, the graph G_t is fixed for time step t following R, while in the second stage, the agent moves to

a vertex in $N_t[v] = \{v\} \cup \{u \in V : (v,u) \in E_t\}$. Notice that, since G is connected, then the cover time under *EUE* is finite, since R models edge-specific delays.

3.3 Cover Time with Zero-Step History

We hereby analyze the cover time of G under *EUE* in the special case when no history is taken into consideration for computing the probability that a given edge is alive at the current time step. Intuitively, each edge appears with a fixed probability p at every time step independently of the others. More formally, for all $e \in E$ and time steps t, $\Pr(e_t = 1) = p \in (0, 1]$.

3.3.1 Random Walk with a Delay

A first approach toward covering G with a single agent is the following: The agent is randomly walking G as if all edges were present and, when an edge is not present, it just waits for it to appear in a following time step. More formally, suppose the agent arrives on a vertex $v \in V$ with (static) degree $d(v)$ at the second stage of time step t. Then, after the graph is fixed for time step $t+1$, the agent selects a neighbor of v, say $u \in N(v)$, uniformly at random, i.e., with probability $\frac{1}{d(v)}$. If $(v,u) \in E_{t+1}$, then the agent moves to u and repeats the above procedure. Otherwise, it remains on v until the first time step $t' > t+1$ such that $(v,u) \in E_{t'}$ and then moves to u. This way, p acts as a *delay* probability, since the agent follows the same random walk it would on a static graph, but with an expected delay of $\frac{1}{p}$ time steps at each vertex. Notice that, in order for such a strategy to be feasible, each vertex must maintain knowledge about its neighbors in the underlying graph; not just the currently alive ones. From now on, we refer to this strategy for the agent as the *Random Walk with a Delay* (shortly *RWD*).

Now, let us upper bound the cover time of *RWD* by exploiting its strong correlation to a simple random walk on the underlying graph G via Wald's Equation (Theorem 5). Below, let C_G stand for the cover time of a simple random walk on the static graph G.

Theorem 5 ([110]). *Let $X_1, X_2, \ldots, X_N$ be a sequence of real-valued, independent and identically distributed random variables where N is a nonnegative integer random variable independent of the sequence (in other words, a stopping time for the sequence). If each X_i and N have finite expectations, then it holds*

$$E[X_1 + X_2 + \ldots + X_N] = E[N] \cdot E[X_1]$$

Theorem 6. *For any connected underlying graph G evolving under the zero-step history* EUE*, the cover time for* RWD *is expectedly C_G/p.*

Proof. Consider a simple random walk, shortly *SRW*, and an *RWD* (under the *EUE* model) taking place on a given connected graph G. Given that *RWD* decides on the next vertex

to visit uniformly at random based on the underlying graph, that is, in exactly the same way *SRW* does, we use a coupling argument to enforce *RWD* and *SRW* to follow the exact same trajectory, i.e., sequence of visited vertices.

Then, let the trajectory end when each vertex in G has been visited at least once and denote by T the total number of vertex transitions made by the agent. Such a trajectory under *SRW* will cover all vertices in expectedly $E[T] = C_G$ time steps. On the other hand, in the *RWD* case, for each transition we have to take into account the delay experienced until the chosen edge becomes available. Let $D_i \geq 1$ be a random variable, where $1 \leq i \leq T$ stands for the actual delay corresponding to vertex transition i in the trajectory. Then, the expected number of time steps till the trajectory is realized is given by $E[D_1 + \ldots + D_T]$. Since the random variables D_i are independent and identically distributed by the edge-uniformity of our model, T is a stopping time for them and all of them have finite expectations, then by Theorem 5 we get $E[D_1 + \ldots + D_T] = E[T] \cdot E[D_1] = C_G \cdot 1/p$. □

For an explicit general bound on *RWD*, it suffices to use $C_G \leq 2m(n-1)$ proved in [3].

A Modified Electrical Network. Another way to analyze the above procedure is to make use of a modified version of the standard literature approach of electrical networks and random walks [30, 50]. That is, we hereby (in Lemmata 12, 13 and Theorem 7) provide a generalization of the results given in [30, 50] thus correlating the hitting and commute times of *RWD* to an electrical network analog and reaching a conclusion for the cover time similar to the one of Theorem 6. This point of view is interesting in its own respect since it provides a first way to perceive electrical network theory in a more general fashion. Ideally, in future work, we would like to generalize it in different ways to capture various random walk models in dynamic environments.

In particular, given the underlying graph G, we design an electrical network, $N(G)$, with the same edges as G, but where each edge has a resistance of $r = \frac{1}{p}$ ohms. Let $H_{u,v}$ stand for the hitting time from vertex u to vertex v in G, i.e. the expected number of time steps until the agent reaches v after starting from u and following *RWD*. Furthermore, let $\phi_{u,v}$ declare the electrical potential difference between vertices u and v in $N(G)$ when, for each $w \in V$, we inject $d(w)$ amperes of current into w and withdraw $2m$ amperes of current from a single vertex v. We now upper-bound the cover time of G under *RWD* by correlating $H_{u,v}$ to $\phi_{u,v}$.

Lemma 12. *For all $u, v \in V$, $H_{u,v} = \phi_{u,v}$ holds.*

Proof. Let us denote by C_{uw} the current flowing between two neighboring vertices u and w. Then, $d(u) = \sum_{w \in N(u)} C_{uw}$ since at each vertex the total inward current must match the total outward current (Kirchhoff's first law). Moving forward, $C_{uw} = \phi_{uw}/r = \phi_{uw}/(1/p) = p \cdot \phi_{uw}$ by Ohm's law. Finally, $\phi_{uw} = \phi_{uv} - \phi_{wv}$ since the sum of electrical potential differences forming a path is equal to the total electrical potential difference of the path

(Kirchhoff's second law). Overall, we can rewrite $d(u) = \sum_{w \in N(u)} p(\phi_{u,v} - \phi_{w,v}) = d(u) \cdot p \cdot \phi_{u,v} - p \sum_{w \in N(u)} \phi_{w,v}$. Rearranging gives

$$\phi_{u,v} = \frac{1}{p} + \frac{1}{d(u)} \sum_{w \in N(u)} \phi_{w,v}.$$

Regarding the hitting time from u to v, we rewrite it based on the first step:

$$H_{u,v} = \frac{1}{p} + \frac{1}{d(u)} \sum_{w \in N(u)} H_{w,v}$$

since the first addend represents the expected number of steps for the selected edge to appear due to *RWD*, and the second addend stands for the expected time for the rest of the walk.

Wrapping it up, since both formulas above hold for each $u \in V \setminus \{v\}$, therefore inducing two identical linear systems of n equations and n variables, it follows that there exists a unique solution to both of them and $H_{u,v} = \phi_{u,v}$. □

In the lemma below, let $R_{u,v}$ stand for the *effective resistance* between u and v, i.e., the electrical potential difference induced when flowing a current of one ampere from u to v.

Lemma 13. *For all $u, v \in V$, $H_{u,v} + H_{v,u} = 2mR_{u,v}$ holds.*

Proof. Similarly to the definition of $\phi_{u,v}$ above, one can define $\phi_{v,u}$ as the electrical potential difference between v and u when $d(w)$ amperes of current are injected into each vertex w and $2m$ of them are withdrawn from vertex u. Next, note that changing all currents' signs leads to a new network where for the electrical potential difference, namely ϕ', it holds $\phi'_{u,v} = \phi_{v,u}$. We can now apply the Superposition Theorem (see Section 13.3 in [17]) and linearly superpose the two networks implied from $\phi_{u,v}$ and $\phi'_{u,v}$ creating a new one where $2m$ amperes are injected into u, $2m$ amperes are withdrawn from v and no current is injected or withdrawn at any other vertex. Let $\phi''_{u,v}$ stand for the electrical potential difference between u and v in this last network. By the superposition argument, we get $\phi''_{u,v} = \phi_{u,v} + \phi'_{u,v} = \phi_{u,v} + \phi_{v,u}$, while from Ohm's law we get $\phi''_{u,v} = 2m \cdot R_{u,v}$. The proof concludes by combining these two observations and applying Lemma 12. □

Theorem 7. *For any connected underlying graph G evolving under the zero-step history* EUE*, the cover time for* RWD *is at most $2m(n-1)/p$.*

Proof. Consider a spanning tree T of G. An agent, starting from any vertex, can visit all vertices by performing an Eulerian tour on the edges of T (crossing each edge twice). This is a feasible way to cover G and thus the expected time for an agent to finish the above task provides an upper bound on the cover time. The expected time to cover each edge twice is given by $\sum_{(u,v) \in E_T} (H_{u,v} + H_{v,u})$ where E_T is the edge-set of T with $|E_T| = n - 1$. By Lemma 13, this is equal to $2m \sum_{(u,v) \in E_T} R_{u,v} = 2m \sum_{(u,v) \in E_T} \frac{1}{p} = 2m(n-1)/p$. □

3.3.2 Random Walk on what's Available

Random Walk with a Delay does provide a nice connection to electrical network theory. However, depending on p, there could be long periods of time where the agent is simply standing still on the same vertex. Since the walk is random anyway, waiting for an edge to appear may not sound very wise. Hence, we now analyze the strategy of a *Random Walk on what's Available* (shortly *RWA*). That is, suppose the agent has just arrived at a vertex v after the second stage at time step t and then E_{t+1} is fixed after the first stage at time step $t+1$. Now, the agent picks uniformly at random only amongst the alive incident edges at time step $t+1$. Let $d_{t+1}(v)$ stand for the degree of vertex v in G_{t+1}. If $d_{t+1}(v) = 0$, then the agent does not move at time step $t+1$. Otherwise, if $d_{t+1}(v) > 0$, the agent selects an alive incident edge each having probability $\frac{1}{d_{t+1}(v)}$. The agent then follows the selected edge to complete the second stage of time step $t+1$ and repeats the strategy. In a nutshell, the agent keeps moving randomly on available edges and only remains on the same vertex if no edge is alive at the current time step. Below, let $\delta = \min_{v \in V} d(v)$ and $\Delta = \max_{v \in V} d(v)$.

Theorem 8. *For any connected underlying graph G with min-degree δ, and max-degree Δ, evolving under the zero-step history* EUE*, the cover time for* RWA *is at least $C_G/(1-(1-p)^{\Delta})$ and at most $C_G/(1-(1-p)^{\delta})$.*

Proof. Suppose the agent follows *RWA* and has reached vertex $u \in V$ after time step t. Then, G_{t+1} becomes fixed and the agent selects uniformly at random a neighboring edge to move to. Let M_{uv} (where $v \in \{w \in V : (u,w) \in E\}$) stand for a random variable taking value 1 if the agent moves to vertex v and 0 otherwise. For $k = 1, 2, \ldots, d(u) = d$, let A_k stand for the event that $d_{t+1}(u) = k$. Therefore, $\Pr(A_k) = \binom{d}{k} p^k (1-p)^{d-k}$ is exactly the probability k out of the d edges exist since each edge exists independently with probability p. Now, let us consider the probability $\Pr(M_{uv} = 1 | A_k)$: the probability v will be reached given that k neighbors are present. This is exactly the product of the probability that v is indeed in the chosen k-tuple (say p_1) and the probability that then v is chosen uniformly at random (say p_2) from the k-tuple. $p_1 = \binom{d-1}{k-1}/\binom{d}{k} = \frac{k}{d}$ since the model is edge-uniform and we can fix v and choose any of the $\binom{d-1}{k-1}$ k-tuples with v in them out of the $\binom{d}{k}$ total ones. On the other hand, $p_2 = \frac{1}{k}$ by uniformity. Overall, we get $\Pr(M_{uv} = 1 | A_k) = p_1 \cdot p_2 = \frac{1}{d}$. We can now apply the total probability law to calculate

$$\Pr(M_{uv} = 1) = \sum_{k=1}^{d} \Pr(M_{uv} = 1 | A_k) \Pr(A_k) = \frac{1}{d} \sum_{k=1}^{d} \binom{d}{k} p^k (1-p)^{d-k} = \frac{1}{d}(1-(1-p)^d)$$

To conclude, let us reduce *RWA* to *RWD*. Indeed, in *RWD* the equivalent transition probability is $\Pr(M_{uv} = 1) = \frac{1}{d}p$, accounting both for the uniform choice and the delay p. Therefore, the *RWA* probability can be viewed as $\frac{1}{d}p'$ where $p' = (1-(1-p)^d)$. To achieve edge-uniformity we set $p' = (1-(1-p)^{\delta})$ which lower bounds the delay of each edge and finally we can apply the same *RWD* analysis by substituting p by p'. Similarly, we can

set the upper-bound delay $p'' = (1 - (1 - p)^{\Delta})$ to lower-bound the cover time. Applying Theorem 6 completes the proof. □

The value of δ used to lower-bound the transition probability may be a harsh estimate for general graphs. However, it becomes quite more accurate in the special case of a d-regular underlying graph where $\delta = \Delta = d$. To conclude this section, we provide a worst-case lower bound on the cover time based on similar techniques as above.

Lemma 14. *There exists an underlying graph G evolving under the zero-step history* EUE *such that the* RWA *cover time is at least $\Omega(mn/(1 - (1 - p)^{\Delta}))$.*

Proof. We consider the $L_n^{2n/3}$ lollipop graph which is known to attain a cover time of $\Omega(mn)$ for a simple random walk [25, 52]. Applying the lower bound from Theorem 8 completes the proof. □

3.4 Cover Time with One-Step History

We now turn our attention to the case where the current state of an edge affects its next state. That is, we take into account a history of length one when computing the probability of existence for each edge independently. A Markovian model for this case was introduced in [36]; see Table 3.1. The left side of the table accounts for the current state of an edge, while the top for the next one. The respective table box provides us with the probability of transition from one state to the other. Intuitively, another way to refer to this model is as the *Birth-Death* model: a dead edge becomes alive with probability p, while an alive edge dies with probability q.

Table 3.1: Birth-Death chain for a single edge [36]

	dead	*alive*
dead	$1 - p$	p
alive	q	$1 - q$

Let us now consider an underlying graph G evolving under the *EUE* model where each possible edge independently follows the aforementioned stochastic rule of evolution.

3.4.1 RWD for General (p, q)-Graphs

Let us hereby derive some first bounds for the cover time of *RWD* via a min-max approach. The idea here is to make use of the "being alive" probabilities to prove lower and upper bounds for the cover time parameterized by $\xi_{min} = \min\{p, 1-q\}$ and $\xi_{max} = \max\{p, 1-q\}$.

Let us consider an *RWD* walk on a general connected graph G evolving under *EUE* with a zero-step history rule dictating $\Pr(e_t = 1) = \xi_{min}$ for any edge e and time step t. We

refer to this walk as the *Upper Walk with a Delay*, shortly *UWD*. Respectively, we consider an *RWD* walk when the stochastic rule of evolution is given by $\Pr(e_t = 1) = \xi_{max}$. We refer to this specific walk as the *Lower Walk with a Delay*, shortly *LWD*. Below, we make use of *UWD* and *LWD* in order to bound the cover time of *RWD* in general (p, q)-graphs.

Theorem 9. *For any connected underlying graph G and the Birth-Death rule, the cover time of* RWD *is at least C_G/ξ_{max} and at most C_G/ξ_{min}.*

Proof. Regarding *UWD*, one can design a corresponding electrical network where each edge has a resistance of $1/\xi_{min}$ capturing the expected delay till any possible edge becomes alive. Applying Theorem 6, gives a C_G/ξ_{min} upper bound for the *UWD* cover time.

Let C' stand for the *UWD* cover time and C stand for the cover time of *RWD* under the Birth-Death rule. It now suffices to show $C \leq C'$ to conclude.

In Birth-Death, the expected delay before each edge traversal is either $1/p$, in case the possible edge is dead, or $1/(1-q)$, in case the possible edge is alive. In both cases, the expected delay is upper-bounded by the $1/\xi_{min}$ delay of *UWD* and therefore $C \leq C'$ follows since any trajectory under *RWD* will take at most as much time as the same trajectory under *UWD*.

In a similar manner, the cover time of *LWD* lower bounds the cover time of *RWD* and, by applying Theorem 6, we derive a lower bound of C_G/ξ_{max}. □

3.5 An Exact Approach

So far, we have established upper and lower bounds for the *cover time* of edge-uniform stochastically-evolving graphs, i.e., the expected time until each vertex is visited at least once by the agent. Our bounds are based on combining extended results from simple random walk theory and careful delay estimations. In this section, we describe an approach to determine the *exact* value of the cover time for temporal graphs evolving under *any* stochastic model. Then, we apply this approach to the already seen zero-step history and one-step history cases of *RWA*.

The key component of our approach is a Markov chain capturing both phases of evolution: the graph dynamics and the walk trajectory. In that case, calculating the cover time reduces to calculating the hitting time to a particular subset of Markov states. Although computationally intractable for large graphs, such an approach provides the exact cover time value and is hence practical for smaller graphs.

Suppose we are given an underlying graph $G = (V, E)$ and a set of stochastic rules R capturing the evolution dynamics of G. That is, R can be seen as a collection of probabilities of transition from one graph instance to another. We denote by k the (longest) history length taken into account by the stochastic rules. Like before, let $n = |V|$ stand for the number of vertices and $m = |E|$ for the number of possible edges of G. We define a Markov chain M with states of the form (H, v, V_c), where

- $H = (H_1, H_2, \ldots, H_k)$, is a k-tuple of *temporal graph instances*, that is, for each $i = 1, 2, \ldots, k$, H_i is the graph instance present $i - 1$ time steps before the current one (which is H_1)
- $v \in V(G)$ is the current position of the agent
- $V_c \subseteq V(G)$ is the set of already covered vertices, i.e., the set of vertices which have been visited at least once by the agent

As described earlier for our edge-uniform model, we assume evolution happens in two phases. First, the new graph instance is determined according to the rule-set R. Second, the new agent position is determined based on a random walk on what's available. In this respect, consider a state $S = (H, v, V_c)$ and another state $S' = (H', v', V_c')$ of the described Markov chain M. Let $\Pr[S \to S']$ denote the transition probability from S to S'. We seek to express this probability as a product of the probabilities for the two phases of evolution. The latter is possible, since, in our model, the random walk strategy is independent of the graph evolution.

For the graph dynamics, let $\Pr[H \xrightarrow{R} H']$ stand for the probability to move from a history-tuple H to another history-tuple H' under the rules of evolution in R. Note that, for $i = 1, 2, \ldots, k-1$, it must hold $H'_{i+1} = H_i$ in order to properly maintain history, otherwise the probability becomes zero. On the other hand, for valid transitions, the probability reduces to $\Pr[H'_1 | (H_1, H_2, \ldots, H_k)]$, which is exactly the probability that H'_1 becomes the new instance given the history $H = (H_1, H_2, \ldots, H_k)$ of past instances (and any such probability is either given directly or implied by R).

For the second phase, i.e., the random walk on what's available, we denote by $\Pr[v \xrightarrow{H_j} v']$ the probability of moving from v to v' on some graph instance H_j. Since, the random walk strategy is only based on the current instance, we can derive a general expression for this probability, which is independent of the graph dynamics R. Below, let $N_{H_j}(v)$ stand for the set of neighbors of v in graph instance H_j. If $\{v, v'\} \notin E(G)$, that is, if there is no possible edge between v and v', then for any temporal graph instance H_j, it holds $\Pr[v \xrightarrow{H_j} v'] = 0$. The probability is also zero for all graph instances H_j where the possible edge is not alive, i.e., $\{v, v'\} \notin E(H_j)$. In contrast, if $\{v, v'\} \in E(H_j)$, then $\Pr[v \xrightarrow{H_j} v'] = |N_{H_j}(v)|^{-1}$, since the agent chooses a destination uniformly at random out of the currently alive ones. Finally, if $v = v'$, then the agent remains still, with probability 1, only if there exist no alive incident edges. We summarize the above facts in the following:

$$\Pr[v \xrightarrow{H_j} v'] = \begin{cases} 1 & \text{, if } N_{H_j}(v) = \emptyset \text{ and } v' = v \\ |N_{H_j}(v)|^{-1} & \text{, if } v' \in N_{H_j}(v) \\ 0 & \text{, otherwise} \end{cases} \tag{3.1}$$

Overall, we combine the two phases in M and introduce the following probabilities.

- If $|V_c|< n$:

$$\Pr[(H,v,V_c)\to(H',v',V_c')] = \begin{cases} \Pr[H \xrightarrow{R} H']\cdot\Pr[v \xrightarrow{H_1'} v'] & \text{, if } v'\in V_c' \text{ and } V_c'=V_c \\ \Pr[H \xrightarrow{R} H']\cdot\Pr[v \xrightarrow{H_1'} v'] & \text{, if } v'\neq v, v'\notin V_c' \text{ and } V_c'=V_c\cup\{v'\} \\ 0 & \text{, otherwise} \end{cases}$$

- If $|V_c|= n$:

$$\Pr[(H,v,V_c)\to(H',v',V_c')] = \begin{cases} 1 & \text{, if } H=H', v=v', V_c=V_c' \\ 0 & \text{, otherwise} \end{cases}$$

For $|V_c|< n$, notice that only two cases may have a non-zero probability with respect to the growth of V_c. If the newly visited vertex v' is already covered, then V_c' must be identical to V_c since no new vertices are covered during this transition. Further, if a new vertex v' is not yet covered, then V_c' is updated to include it as well as all the covered vertices in V_c.

For $|V_c|= n$, the idea is that once such a state has been reached, and so all vertices are covered, then there is no need for further exploration. Therefore, such a state can be made *absorbing*. In this respect, let us denote the set of these states as $\Gamma = \{(H,v,V_c)\in M \ : |V_c|= n\}$.

Definition 6. *Let* $\mathrm{ECT}(G,R)$ *be the problem of determining the exact value of the cover time for an* RWA *on a graph* G *stochastically evolving under rule-set* R*.*

Theorem 10. *Assume all probabilities of the form* $\Pr[H \xrightarrow{R} H']$ *used in* M *are exact reals and known a priori. Then, for any underlying graph* G *and stochastic rule-set* R*, it holds that* $\mathrm{ECT}(G,R)\in$ EXPTIME*.*

Proof. For each temporal graph instance, H_i, in the worst case, there exist 2^m possibilities, since each of the m possible edges is either alive or dead at a graph instance. For the whole history H, the number of possibilities becomes $(2^m)^k = 2^{k\cdot m}$ by taking the product of k such terms. There are n possibilities for the walker's position v. Finally, for each $v\in V(G)$, we only allow states such that $v\in V_c$. Therefore, since we fix v, there are up to $n-1$ vertices to be included or not in V_c leading to a total of $\mathcal{O}(2^{n-1})$ possibilities for V_c. Taking everything into account, M has a total of $\mathcal{O}(2^{k\cdot m+n-1}n)$ states.

Let $H_{s,\Gamma}$ stand for the hitting time of Γ when starting from a state $s\in M$. Assuming exact real arithmetic, we can compute all such hitting times by solving the following system (Theorem 1.3.5 [98]):

$$\begin{cases} H_{s,\Gamma} = 0 & ,\forall s\in\Gamma \\ H_{s,\Gamma} = 1+\sum_{s'\notin\Gamma}\Pr[s\to s']\cdot H_{s',\Gamma} & ,\forall s\notin\Gamma \end{cases}$$

Let C stand for the cover time of an *RWA* on G evolving under R. By definition, the cover time is the expected time till all vertices are covered, regardless of the position of the

walker at that time. Consider the set $S = \{(H, v, \{v\}) \in M : v \in V(G)\}$ of start positions for the agent as depicted in M. Then, it follows $C = \max_{s \in S} H_{s,\Gamma}$, where we take the worst-case hitting time to a state in Γ over any starting position of the agent. In terms of time complexity, computing C requires computing all values $H_{s,\Gamma}$, for every $s \in S$. To do so, one must solve the above linear system of size $\mathcal{O}(2^{k \cdot m+n-1}n)$, which can be done in time exponential to input parameters n, m and k. □

It's noteworthy to remark that this approach is general in the sense that there are no assumptions on the graph evolution rule-set R besides it being stochastic, i.e., describing the probability of transition from each graph instance to another given some history of length k. In this regard, Theorem 10 captures both the case of Markovian Evolving Graphs [7] and the case of Edge-Uniform Graphs considered in this paper. We now proceed and show how the aforementioned general approach applies to the zero-step and one-step history cases of Edge-Uniform Graphs. To do so, we calculate the corresponding graph-dynamics probabilities. The random walk probabilities were already given in Equation 3.1.

RWA on Edge-Uniform Graphs (Zero-Step History). Based on the general model, we rewrite the transition probabilities for the special case when *RWA* takes place on an edge-uniform graph without taking into account any memory, i.e., the same case as in Section 3.3. Notice that, since past instances are not considered in this case, the history-tuple reduces to a single graph instance H. We rewrite the transition probabilities, for the case $|V_c| < n$, as follows:

$$\Pr[(H, v, V_c) \to (H, v', V_c')] = \begin{cases} \Pr[H'|H] \cdot \Pr[v \xrightarrow{H'} v'] & \text{, if } v' \in V_c' \text{ and } V_c' = V_c \\ \Pr[H'|H] \cdot \Pr[v \xrightarrow{H'} v'] & \text{, if } v' \neq v, v' \notin V_c' \text{ and } V_c' = V_c \cup \{v'\} \\ 0 & \text{, otherwise} \end{cases}$$

Let α stand for the number of edges alive in H'. Since there is no dependence on history and each edge appears independently with probability p, we get $\Pr[H'|H] = \Pr[H'] = p^{\alpha} \cdot (1-p)^{m-\alpha}$.

RWA on Edge-Uniform Graphs (One-Step History). We hereby rewrite the transition probabilities for a Markov chain capturing an *RWA* taking place on an edge-uniform graph where, at each time step, the current graph instance is taken into account to generate the next one. This case is related to the results in Section 3.4. Due to the history inclusion, the transition probabilities become more involved than those seen for the zero-history case. Again, we consider the non-absorbing states, where $|V_c| < n$.

$$\Pr[((H_1,H_2),v,V_c)\to((H_1',H_2'),v',V_c')]=\begin{cases}\Pr[(H_1,H_2)\to(H_1',H_2')]\cdot\Pr[v\xrightarrow{H_1'}v'] & \text{, if } v'\in V_c' \text{ and } V_c'=V_c\\ \Pr[(H_1,H_2)\to(H_1',H_2')]\cdot\Pr[v\xrightarrow{H_1'}v'] & \text{, if } v'\notin V_c' \text{ and } V_c'=V_c\cup\{v'\}\\ 0 & \text{, otherwise}\end{cases}$$

If $H_2' \neq H_1$, i.e., if it does not hold that, for each $e \in G$, $e \in H_2'$ if and only if $e \in H_1$, then $\Pr[(H_1,H_2)\to(H_1',H_2')]=0$, otherwise the history is not properly maintained. On the other hand, if $H_2'=H_1$, then $\Pr[(H_1,H_2)\to(H_1',H_2')]=\Pr[(H_1,H_2)\to(H_1',H_1)]=\Pr[H_1'|H_1]$. To derive an expression for the latter, we need to consider all edge (mis)matches between H_1' and H_1, and properly apply the Birth-Death rule (Table 3.1). Below, we denote by $D(H)=E(G)\setminus E(H)$ the set of possible edges of G, which are dead at instance H. Let $c_{00}=|D(H_1)\cap D(H_1')|$, $c_{01}=|D(H_1)\cap E(H_1')|$, $c_{10}=|E(H_1)\cap D(H_1')|$ and $c_{11}=|E(H_1)\cap E(H_1')|$. Each of the c_{00} edges was dead in H_1 and remained dead in H_1', with probability $1-p$. Similarly, each of the c_{01} edges was dead in H_1 and became alive in H_1', with probability p. Also, each of the c_{10} edges was alive in H_1 and died in H_1', with probability q. Finally, each of the c_{11} edges was alive in H_1 and remained alive in H_1', with probability $1-q$. Overall, due to the edge-independence of the model, we get $\Pr[H_1'|H_1]=(1-p)^{c_{00}}\cdot p^{c_{01}}\cdot q^{c_{10}}\cdot(1-q)^{c_{11}}$.

3.6 Concluding Remarks

We defined the general *Edge-Uniform Evolution* model for a stochastically-evolving graph, where a single stochastic rule is applied, but to each edge independently, and provided lower and upper bounds for the cover time of two random walks taking place on such a graph (cases $k=0,1$). Moreover, we provided a general framework to compute the exact cover time of a broad family of stochastically-evolving graphs in exponential time.

An immediate open problem is to obtain a good bound for the cover time of *RWA* in the Birth-Death model. In this case, the problem becomes more complex than the $k=0$ case. Depending on the values of p and q, the walk may be heavily biased, positively or negatively, toward possible edges incident to the walker's position, which were used in the recent past. Another idea is to try proving cover time bounds *with high probability* rather than focusing on the expected value. A more careful probabilistic analysis together with useful tools, e.g., Chernoff bounds, might be pertinent in this case.

Our model seems to be on the opposite end of the Markovian evolving graph model [7]. There, the evolution of possible edges directly depends on the family of graphs selected as possible instances. So, a research direction we suggest is to devise another model of *partial* edge-dependency. That is, we wish the edge-specific stochastic rule to depend on a proper subset of the edge-set; neither on no other edge nor on every other edge. Such a model may prove interesting in terms of community-partitioned networks or block-defined graphs.

Chapter 4

Eternal Domination

4.1 Introduction

As a natural goal in military deterrence and defence strategies, patrolling a network has always remained topical throughout history. In the context of graph searching, such a patrolling task is often modeled as a combinatorial pursuit-evasion game played on a graph. Herein, we study such a game for a task that requires the eternal domination of a network.

The *Roman Domination* problem was introduced in [108]: where should Emperor Constantine the Great have located his legions in order to optimally defend against attacks in unsecured locations without leaving another location unsecured? In graph theoretic terms, the interest is in producing a *dominating set* of the graph, i.e., a guard placement where each vertex must have a guard on it or on at least one of its neighbors, with possibly some extra problem-specific qualities. Some seminal work on this topic includes [70, 103].

The above model caters only for a single attack on an unsecured vertex. A natural question is to consider special domination strategies against a sequence of attacks on the same graph [27]. In this setting, (some of) the guards are allowed to move after each attack to defend against it and modify their placement. The difficulty here lies in establishing a guards' placement to retain domination after coping with each attack. Such a sequence of attacks can be of finite, i.e., a set of k consecutive attacks, or even *infinite* length.

In this paper, we focus on the latter. We wish to protect a graph against attacks happening indefinitely on its vertices. Initially, the guards are placed on some vertices of the graph such that they form a dominating set, *with at most one guard per vertex*. Then, an attack occurs on an unoccupied vertex. All the guards (may) now move in order to counter the attack: One of them moves to the attacked vertex, while each of the others moves to an adjacent vertex of theirs such that the new guards' placement again forms a dominating set. This takes place ad infinitum.

The attacker's objective is to devise a sequence of attacks, which leads the guards to a non-dominating placement. On the other hand, the guards wish to maintain a sequence of dominating sets without any interruption. The m-*Eternal Domination* problem, studied

in this chapter, deals with determining the minimum number of guards such that they eternally protect the graph in the above fashion. The focus is on rectangular grids, where, to the best of our knowledge, we provide a first general upper bound.

4.1.1 Related Work

Infinite order domination was first considered by Burger et al. [28] as an extension to finite order domination. Later on, Goddard et al. [65] proved some first bounds with respect to other graph-theoretic notions (like independence and clique cover) for the one-guard-moves and all-guards-move cases. The relationship between eternal domination and clique cover is examined more carefully in [6]. There exists a series of other papers with several combinatorial bounds, e.g., see [66, 69, 71, 82].

Regarding grid graphs, Chang [31] gave many strong upper and lower bounds for the domination number. Indeed, Gonçalves et al. [67] proved Chang's construction optimal for rectangular grids where both dimensions are greater or equal to 16. Moving onward to eternal domination, bounds for $3 \times n$ [55, 95], $4 \times n$ [12] and $5 \times n$ [109] grids have been examined, where the bounds are almost tight for $3 \times n$ and exactly tight for $4 \times n$.

Due to the mobility of the guards and the breakdown into alternate turns, one can view eternal domination as a pursuit-evasion combinatorial game in the same context as *Cops & Robber* [19] (k cops try to arrest a single robber), the *Surveillance Game* [59, 63] (a marker deposits marks on vertices such that a surfer cannot ever reach an unmarked vertex), and the *Spy Game* [39, 40], where a spy (faster than the guards) has to always maintain some predefined distance from each guard. In all such games, there are two players who alternately take turns, with one of them pursuing the other possibly indefinitely. An analogous *Eternal Vertex Cover* problem has been considered [58, 80, 81] with attacks occurring on the edges of the graph. In that setting, the guards defend against an attack by traversing the attacked edge, while they move to preserve a vertex cover after each turn. The m-*eviction number* is studied in [79], where attacks occur on the vertices occupied by guards and they have to move to survive, whilst always maintaining a dominating set.

A good overview of the topic can be found in a recent survey on graph protection [83].

4.1.2 Our Result

We make a first step towards answering an open question raised by Klostermeyer and Mynhardt [83]: We show that, in order to ensure m-eternal domination in rectangular grids, only a linear number of extra guards is needed compared to domination.

To obtain this result, we devise an unravelling strategy of successive (counter) clockwise rotations for the guards to eternally dominate an infinite grid. This strategy is referred to as the *Rotate-Square* strategy. Then, we apply the same strategy to finite grids with some extra guards to ensure the boundary remains always guarded. Overall, we show that $\lceil \frac{mn}{5} \rceil + \mathcal{O}(m+n)$ guards suffice to eternally dominate an $m \times n$ grid, for $m, n \geq 16$.

4.1.3 Outline

In Section 4.2, we define the m-*Eternal Domination* game. Later, in Section 4.3, we describe the basic components of the Rotate-Square strategy and prove that it can be used to dominate an infinite grid forever. Later, in Section 4.4 we show how the strategy can be adjusted to eternally dominate finite grids by efficiently handling movements near the boundary and the corners. Finally, in Section 4.5, we cite some concluding remarks.

4.2 Preliminaries

Eternal Domination can be regarded as a combinatorial pursuit-evasion game played on a graph G. There exist two players: one of them controls the *guards*, while the other controls the *attacker*. The game takes place in *rounds*. Each round consists of two *turns*: one for the guards and one for the attacker.

Initially (round 0), the guard tokens are placed such that they form a dominating set on G. Then, without loss of generality, the attacker attacks a vertex without a guard on it. A guard, dominating the attacked vertex, must now move on it to counter the attack. Notice that at least one such guard exists because their initial placement is dominating. Moreover, the rest of the guards may move; a guard on vertex v can move to any vertex in $N[v]$. The guards must ensure their modified placement is still a dominating set for G. The game proceeds in similar fashion in any subsequent round. Guards win if they can counter any attack of the attacker and eternally maintain a dominating set; that is, for an infinite number of attacks. Otherwise, the attacker wins, as he manages to force the guards to reach a placement that is no longer dominating; then, an attack on an undominated vertex suffices to win. From now on, we say that a vertex is *unoccupied* when no guard lies on it.

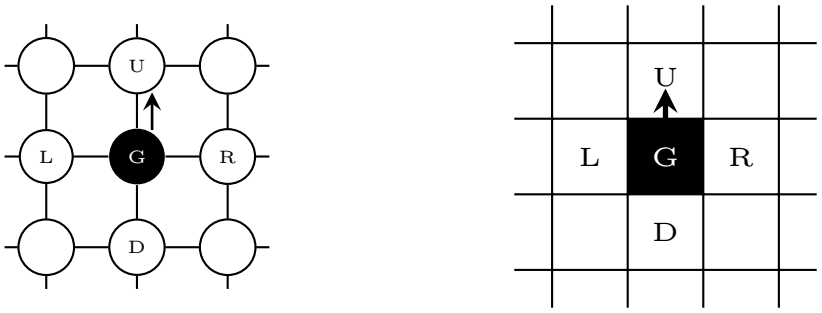
Definition 7. $\gamma_{\mathrm{m}}^{\infty}(G)$ *stands for the* m-eternal domination number *of a graph G, i.e., the minimum size of a guards' team needed to eternally dominate G (when* all *guards can move at each turn).*

Similarly to notation for standard domination, we simplify $\gamma_{\mathrm{m}}^{\infty}(P_m \square P_n)$ to $\gamma_{m,n}^{\infty}$. Since the initial guards' placement is dominating, we get $\gamma_{\mathrm{m}}^{\infty}(G) \geq \gamma(G)$ for any graph G. By a simple rotation, we get $\gamma_{m,n} = \gamma_{n,m}$ and $\gamma_{m,n}^{\infty} = \gamma_{n,m}^{\infty}$. Finally, multiple guards are *not* allowed to lie on a single vertex, since this could provide an advantage for the guards [56].

4.3 Eternally Dominating an Infinite Grid

In this section, we describe a strategy to eternally dominate an *infinite grid*. We denote an infinite grid as G_∞ and define it as a pair $(V(G_\infty), E(G_\infty))$, where $V(G_\infty) = \{(x, y) : x, y \in \mathbb{Z}\}$ and any vertex $(x, y) \in V(G_\infty)$ is adjacent to $(x, y-1)$, $(x, y+1)$, $(x-1, y)$ and $(x+1, y)$. In all figures to follow, we view the grid as a mesh, i.e., similar to a chessboard,

where each *cell* of the mesh corresponds to a vertex of $V(G_\infty)$ and neighbors four other cells: the one above, below, left and right of it. We assume row x is *above* row $x+1$ and column y is *left* of column $y+1$. Each guard occupies a single cell and has the capability to move to an adjacent cell (left, right, up or down) during the guards' turn. For a visual explanation of the grid-mesh correspondence, see Figure 4.1.



(a) Some local view of a grid graph consisting of vertices and edges

(b) The local view of the corresponding mesh configuration consisting of cells

Figure 4.1: From a grid graph (a) to a mesh configuration (b): a guard lying on vertex/cell G can move to any of its neighboring vertices/cells L, R, U, D during the guards' turn

Initially, let us consider a family of dominating sets for G_∞. In the following, let $\mathbb{Z}^2 := \mathbb{Z} \times \mathbb{Z}$ and let $\mathbb{Z}_5 := \{0, 1, 2, 3, 4\}$ stand for the group of integers modulo 5. We then define the function $f : \mathbb{Z}^2 \to \mathbb{Z}_5$ as $f(x, y) = x + 2y \pmod 5$ for any $(x, y) \in \mathbb{Z}^2$. This function appears in [31] and is central to providing an optimal dominating set for sufficiently large finite grids. Intuitively, such an optimal dominating set can be interpreted pictorially as a set of carefully placed "diagonal lines" which perfectly dominate grid neighbors around them (see Figure 4.2a). We therefore try to exploit this function's usefulness in our setting by eternally switching from one such "diagonal lines" dominating set to another. With respect to this goal, let $D_t = \{(x, y) \in V(G_\infty) : f(x, y) = t\}$ for $t \in \mathbb{Z}_5$ and $\mathcal{D}(G_\infty) = \{D_t : t \in \mathbb{Z}_5\}$. For purposes of symmetry, let us define $f'(x, y) = f(y, x)$ and then let $D'_t = \{(x, y) \in V(G_\infty) : f'(x, y) = t\}$ and $\mathcal{D}'(G_\infty) = \{D'_t : t \in \mathbb{Z}_5\}$.

Proposition 5. *Any $D_t, D'_t \in \mathcal{D}(G_\infty) \cup \mathcal{D}'(G_\infty)$ is a dominating set for G_∞.*

Proof. Let $(x, y) \in V(G_\infty)$ and $f(x, y) = t \in \{0, 1, 2, 3, 4\}$. We consider all possible cases for another vertex $(w, z) \in V(G_\infty)$:

- If $f(w, z) = t$, then $(w, z) \in D_t$.
- If $f(w, z) = t + 1 \pmod 5$, then $f(w - 1, z) = t$ and $(w - 1, z) \in D_t$ dominates (w, z).
- If $f(w, z) = t - 1 \pmod 5$, then $f(w + 1, z) = t$ and $(w + 1, z) \in D_t$ dominates (w, z).
- If $f(w, z) = t + 2 \pmod 5$, then $f(w, z - 1) = t$ and $(w, z - 1) \in D_t$ dominates (w, z).
- If $f(w, z) = t - 2 \pmod 5$, then $f(w, z + 1) = t$ and $(w, z + 1) \in D_t$ dominates (w, z).

Similarly, let $(x, y) \in V(G_\infty)$ and $f'(x, y) = t \in \mathbb{Z}_5$. Again, we consider all possible cases for another vertex $(w, z) \in V(G_\infty)$:

- If $f'(w, z) = t$, then $(w, z) \in D'_t$.
- If $f'(w, z) = t+1 \pmod 5$, then $f'(w, z-1) = t$ and $(w, z-1) \in D'_t$ dominates (w, z).
- If $f'(w, z) = t-1 \pmod 5$, then $f'(w, z+1) = t$ and $(w, z+1) \in D'_t$ dominates (w, z).
- If $f'(w, z) = t+2 \pmod 5$, then $f'(w-1, z) = t$ and $(w-1, z) \in D'_t$ dominates (w, z).
- If $f'(w, z) = t-2 \pmod 5$, then $f'(w+1, z) = t$ and $(w+1, z) \in D'_t$ dominates (w, z).

□

Notice that the above constructions form *perfect* dominating sets, i.e., dominating sets where each vertex is dominated by *exactly* one vertex (possibly itself), since, for each vertex $v \in V(G_\infty)$, exactly one vertex from $N[v]$ lies in D_t (respectively D'_t) by the definition of D_t (respectively D'_t).

4.3.1 A First Eternal Domination Strategy

Let us consider a *shifting-style* strategy as a first simple strategy to eternally dominate G_∞. The guards initially pick a placement D_t for some $t \in \mathbb{Z}_5$. Next, an attack occurs on some unoccupied vertex. Since the D_t placement perfectly dominates G_∞, there exists exactly one guard adjacent to the attacked vertex. Therefore, it is mandatory for him to move onto the attacked vertex. His move defines a direction in the grid: left, right, up or down. The rest of the strategy reduces to each guard moving according to the defined direction.

The aforementioned strategy works fine for the infinite grid. Nonetheless, applying it (directly or modified) to a finite grid encounters many obstacles. Shifting the guards toward one course leaves some vertices in the very end of the opposite course (near the boundary) undominated, since there is no longer an unlimited supply of guards to ensure protection. To overcome this problem, we propose a different strategy whose main aim is to redistribute the guards without creating any bias to a specific direction.

4.3.2 Unoccupied Squares

Another m-eternal domination strategy is to *rotate* the guards' placement around subgrids of size 2×2, in which all four cells are unoccupied. We refer to such a subgrid as an *unoccupied square*. Intuitively, by using such an approach, the overall movement is zero and the guards always occupy a placement in $\mathcal{D}(G_\infty) \cup \mathcal{D}'(G_\infty)$ after an attack is defended.

Consider some vertex $(x, y) \in V(G_\infty)$, where $(x, y) \in D_t$ for some value $t \in \mathbb{Z}_5$. Now, assume that the guards lie on the vertices specified in D_t and hence form a dominating set. In Figure 4.2a, we partially view G_∞ where the black cell represents a guard on some cell $(x, y) \in D_t$ and the gray cells represent guards elsewhere in D_t. By looking around (x, y),

we identify the existence of four *unoccupied squares*. For $i = 0, 1, 2, 3$, let $SQ_i(x, y)$ denote the i-th unoccupied square with respect to (x, y).

- $SQ_0(x, y) = \{(x-1, y+1), (x-1, y+2), (x, y+1), (x, y+2)\}$
- $SQ_1(x, y) = \{(x+1, y), (x+1, y+1), (x+2, y), (x+2, y+1)\}$
- $SQ_2(x, y) = \{(x, y-2), (x, y-1), (x+1, y-2), (x+1, y-1)\}$
- $SQ_3(x, y) = \{(x-2, y-1), (x-2, y), (x-1, y-1), (x-1, y)\}$

In Figure 4.2a, a cell in an unoccupied square $SQ_i(x, y)$ has a label SQ_i.

One can verify that, for every $(w, z) \in \bigcup_{i=0}^{3} SQ_i(x, y)$, we get $f(w, z) \neq f(x, y)$ and thus $(w, z) \notin D_t$. Notice that (x, y) has exactly one adjacent cell in each of these unoccupied squares and is the only guard that dominates these four cells, since domination is perfect. We say that a guard on (x, y) *slides along* the side of an unoccupied square $SQ_i(x, y)$ when he moves to cell (w, z) adjacent to (x, y), where (w, z) is also adjacent to a cell in $SQ_i(x, y)$. In other words, the (x, y)-guard's current and previous cells are both adjacent to a cell in $SQ_i(x, y)$. In the case of a D_t placement, see Figure 4.2a, an attack on a cell $(w, z) \in SQ_i(x, y) \cap N((x, y))$ would mean the guard on (x, y) moves to (w, z) and slides along the side of unoccupied square $SQ_{(i+1) \bmod 4}(x, y)$. For example, an attack on the bottom-right cell of SQ_3 would mean the (x, y)-guard slides along SQ_0: Figures 4.2b, 4.2c.

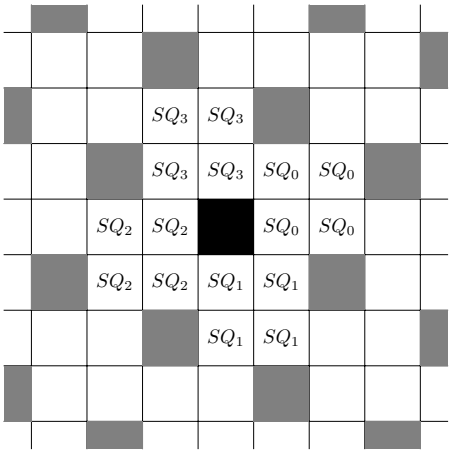
The aforementioned observations also extend to some vertex (x, y) in a dominating set D'_t. We now define the four unoccupied squares as follows (see Figure 4.3a):

- $SQ'_0(x, y) = \{(x, y+1), (x, y+2), (x+1, y+1), (x+1, y+2)\}$
- $SQ'_1(x, y) = \{(x+1, y-1), (x+1, y), (x+2, y-1), (x+2, y)\}$
- $SQ'_2(x, y) = \{(x-1, y-2), (x-1, y-1), (x, y-2), (x, y-1)\}$
- $SQ'_3(x, y) = \{(x-2, y), (x-2, y+1), (x-1, y), (x-1, y+1)\}$

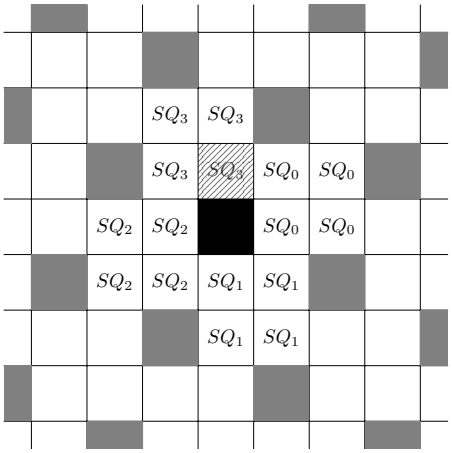
Similarly to before, the squares are unoccupied, since for every $(w, z) \in \bigcup_{i=0}^{3} SQ'_i$ we get $f'(w, z) \neq f'(x, y)$ and thus $(w, z) \notin D'_t$. The (x, y)-guard has exactly one adjacent cell in each of these unoccupied squares and protecting an attack on SQ'_i now means sliding along the side of $SQ'_{(i-1) \bmod 4}$. For example, an attack on the bottom-right cell of SQ'_2 means the (x, y)-guard slides along SQ'_1 (Figures 4.3b, 4.3c).

4.3.3 The Rotate-Square Strategy

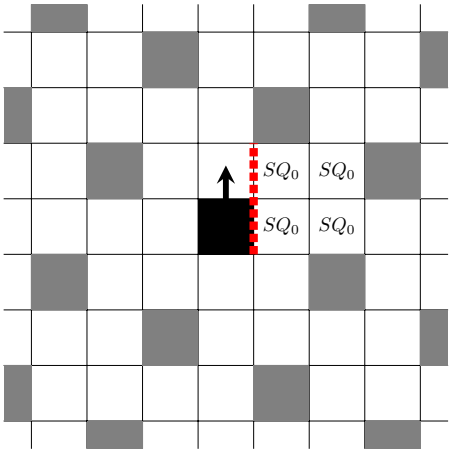
We hereby describe the *Rotate-Square* strategy and prove it eternally dominates G_∞. The strategy makes use of the unoccupied squares idea. Once an attack occurs on some unoccupied vertex, since any D_t or D'_t dominating set the guards form is perfect, there exists a single guard who is able to defend against it. We refer to this guard as the *defence-responsible* guard. Without loss of generality, let the defence-responsible guard lie on some cell $(x, y) \in D_t$ for some t. For D'_t placements, the arguments are similar. We identify the



(a) Unoccupied squares around $(x, y) \in D_t$; (x, y) in black

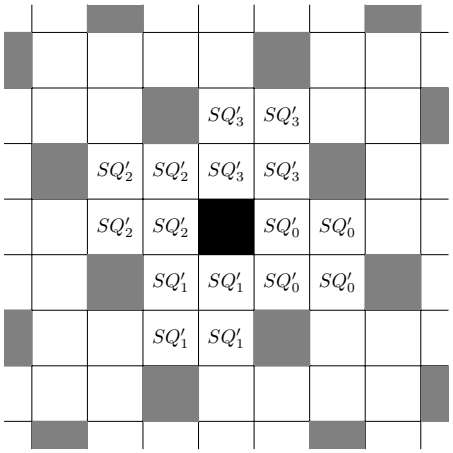


(b) Attack on bottom-right cell of $SQ_3(x, y)$: we identify the defence-responsible guard (x, y) (in black) and the corresponding unoccupied squares

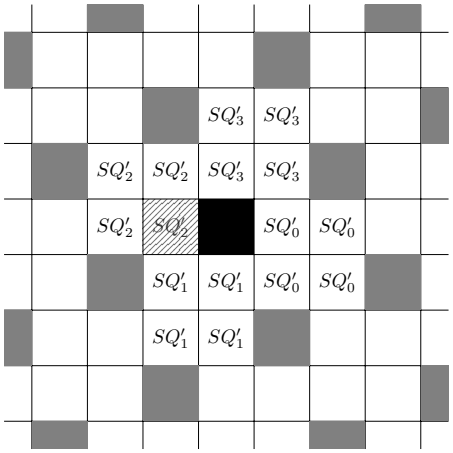


(c) The defence-responsible guard slides along a side of $SQ_0(x, y)$ (dotted line): its current and next cell are both adjacent to $SQ_0(x, y)$ cells

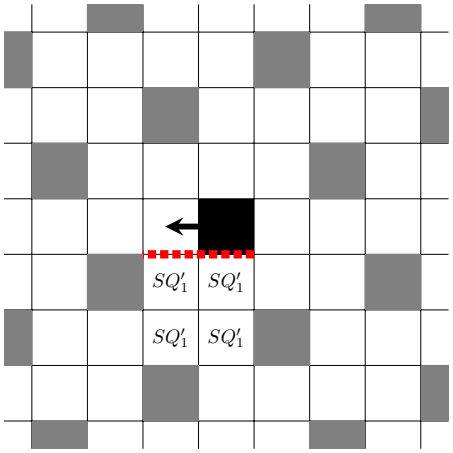
Figure 4.2: Examples of defence-responsible guard, unoccupied squares and sliding along for the D_t case



(a) Unoccupied squares around $(x, y) \in D'_t$; (x, y) in black



(b) Attack on bottom-right cell of $SQ'_2(x, y)$: we identify the defence-responsible guard (x, y) (in black) and the corresponding unoccupied squares



(c) The defence-responsible guard slides along a side of $SQ'_1(x, y)$ (dotted line): its current and next cell are both adjacent to $SQ'_1(x, y)$ cells

Figure 4.3: Examples of defence-responsible guard, unoccupied squares and sliding along for the D'_t case

four unoccupied squares around the defence-responsible guard as in Figure 4.2a. Assume the attack happened on a vertex $(w, z) \in SQ_i(x, y)$. Then, as discussed earlier, the defence-responsible guard moves to (w, z) to defend against the attack and such a move means he is sliding along the side of square $SQ_{(i+1) \bmod 4}(x, y)$. We refer to $SQ_{(i+1) \bmod 4}(x, y)$ as the *pattern square*, i.e., the unoccupied square along whose side the defence-responsible guard slides to defend the attack.

Notice that, due to the grid topology and the fact that a D_t placement is a perfect dominating set, there are exactly four guards adjacent to cells of the pattern square: exactly one guard per cell of the pattern square (one of them being the defence-responsible guard). We refer to these four guards as the *pattern guards*. In Figures 4.4 and 4.5, we identify the pattern guards for each potential pattern square out of the four unoccupied squares related to a D_t or D'_t placement. Besides the defence-responsible guard, the other three guards dominating the pattern square also slide along a side of the pattern square, such that the four guards' overall movement looks as a rotation step around the pattern square.

To facilitate a more formal explanation, let us break down the guards' turn into a few distinct components. Of course, the guards are always assumed to move concurrently during their turn.

Initially, the guards occupy a dominating set D_t in $\mathcal{D}(G_\infty)$. Then, an attack occurs on a vertex in $V(G_\infty) \setminus D_t$. To defend against it, the guards apply *Rotate-Square*:

(1) Identify the defence-responsible guard.

(2) Identify the pattern square and the pattern guards.

(3) The pattern guards slide along the sides of the pattern square.

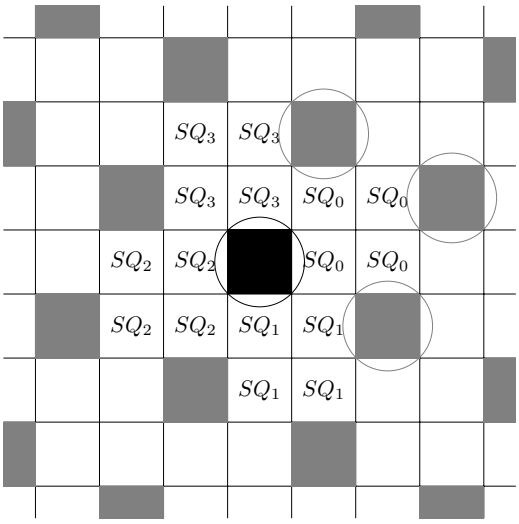
(4) Repeat the rotation pattern in horizontal and vertical lanes in hops of distance five.

Let us examine each of these strategy components more carefully.

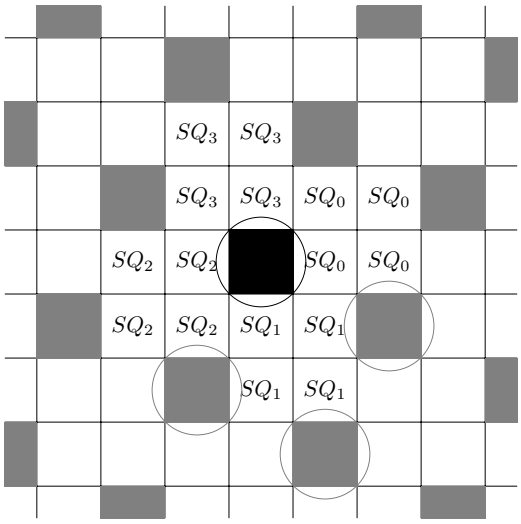
Step (1) requires looking at the grid and spotting the guard, which lies on a cell adjacent to the attacked cell. The four unoccupied squares around the defence-responsible guard are identified as in Figure 4.2a.

In step (2), the pattern square is identified as the unoccupied square along whose side the defence-responsible guard has to slide to defend the attack. The four guards adjacent to cells of the pattern square are identified as the pattern guards.

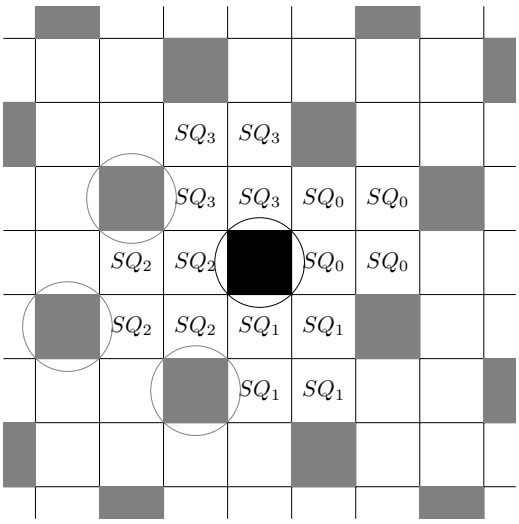
In step (3), each of the pattern guards (including the defence-responsible guard) slides along a side of the pattern square. For an example, see Figure 4.6: the defence-responsible guard in cell (x, y) (in black) defends against an attack on the bottom-right cell of $SQ_3(x, y)$ by sliding along $SQ_0(x, y)$. Then, the other three guards around $SQ_0(x, y)$ (in gray) slide along a side of $SQ_0(x, y)$ as well. The latter happens in order to preserve that the pattern square $SQ_0(x, y)$ remains unoccupied.



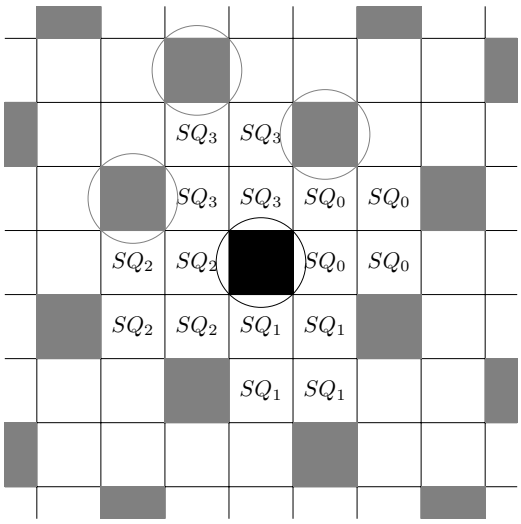
(a) Pattern guards for SQ_0



(b) Pattern guards for SQ_1

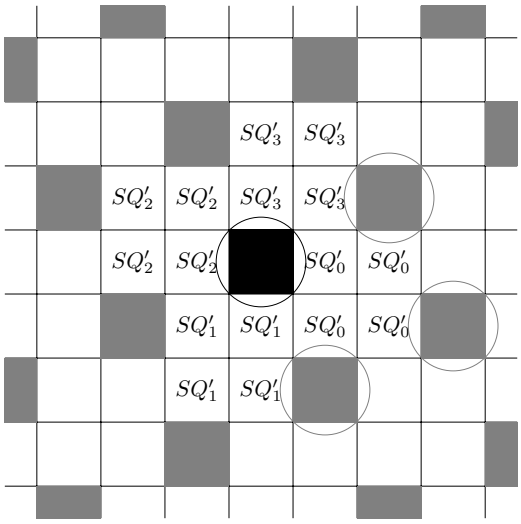


(c) Pattern guards for SQ_2

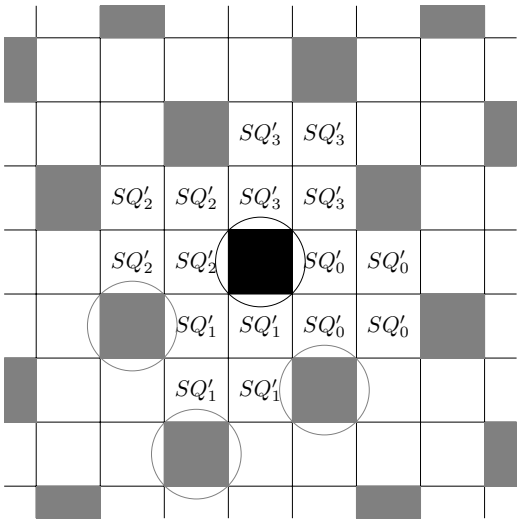


(d) Pattern guards for SQ_3

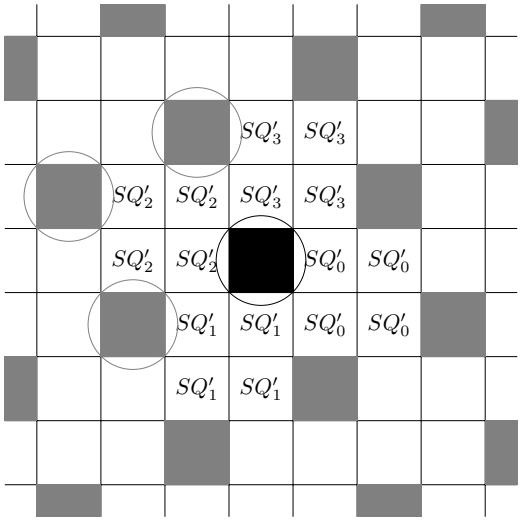
Figure 4.4: Pattern guards for D_t unoccupied squares (circled)



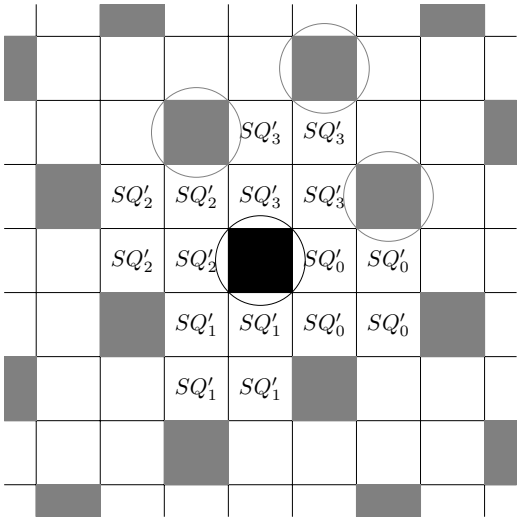
(a) Pattern guards for SQ'_0



(b) Pattern guards for SQ'_1

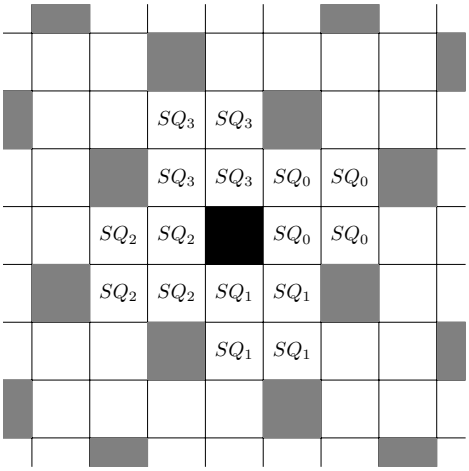


(c) Pattern guards for SQ'_2

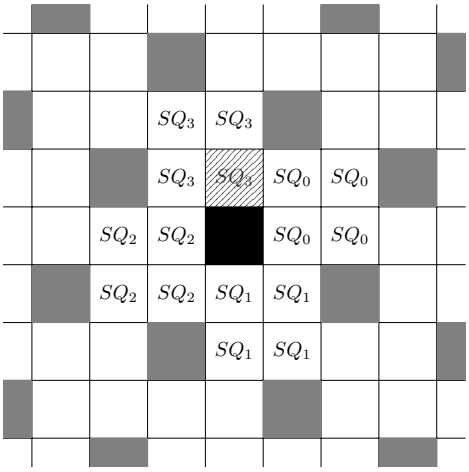


(d) Pattern guards for SQ'_3

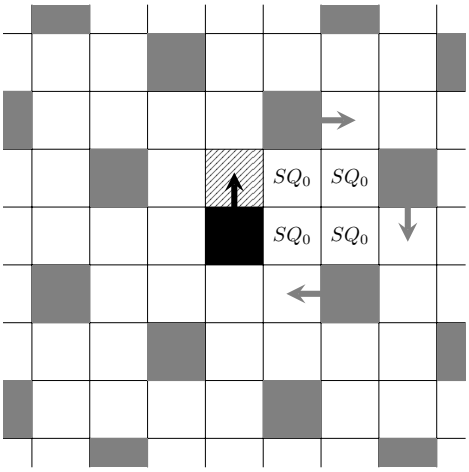
Figure 4.5: Pattern guards for D'_t unoccupied squares (circled)



(a) Unoccupied squares around $(x, y) \in D_t$; (x, y) in black



(b) Attack on bottom-right cell of $SQ_3(x, y)$: the defence-responsible guard in black cell (x, y) must slide along a side of $SQ_0(x, y)$, which is identified as the pattern square for the next guards' turn



(c) Pattern guards slide along sides of pattern square $SQ_0(x, y)$; arrows indicate the directions

Figure 4.6: An example for Step (3) of Rotate-Square

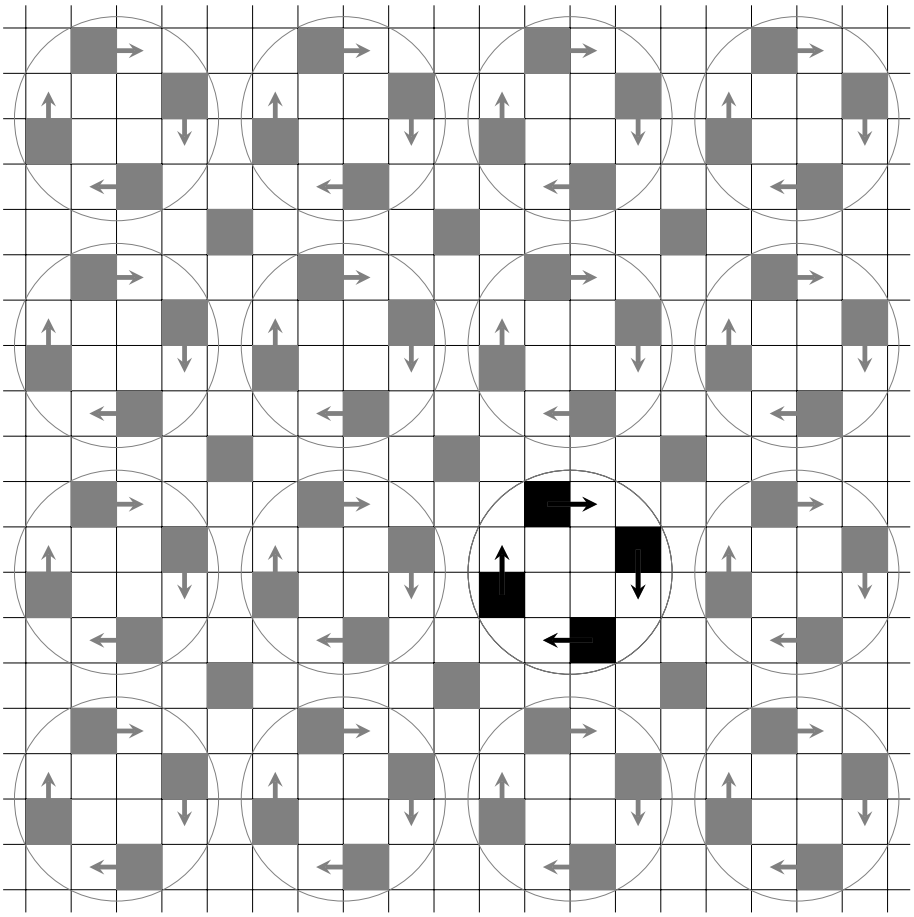


Figure 4.7: Example execution for Step (4) of Rotate-Square: pattern guards in black

Eventually, in step (4), the pattern square is used as a guide for the move of the rest of the guards in D_t. Consider a pattern guard initially lying on vertex (w, z). By construction of D_t, guards lie on all vertices $(w \pm 5\alpha, z \pm 5\beta)$ for $\alpha, \beta \in \mathbb{N}$, since adding multiples of 5 in both dimensions does not affect membership in D_t by definition of $f(\cdot)$. We refer to the set $\{(w \pm 5\alpha, z \pm 5\beta) : \alpha, \beta \in \mathbb{N}\}$ as the *cousins* of (w, z). Each pattern guard, in step (3), slides along a side of the pattern square. His move defines a direction on the grid: up, down, left, right. For each pattern guard (w, z), the strategy of his cousins reduces to taking a step in the same direction. The rest of the guards, i.e., guards who are not cousins to any pattern guard, do not move during this turn, i.e., will remain in their pre-attack location after the attack. From now on, we refer to these guards as *stationary guards*. We provide an example execution of step (4) in Figure 4.7. The original pattern guards are given in black. The circles enclose the repetitions of the pattern guards' move by their cousins, in gray. Guards outside a circle do not move during this turn, i.e., they remain in their pre-attack location.

Lemma 15. *Assume the guards occupy a dominating placement D in $\mathcal{D}(G_\infty) \cup \mathcal{D}'(G_\infty)$ and an attack occurs on a vertex in $V(G_\infty) \setminus D$. After applying Rotate-Square, the guards successfully defend against the attack and again form a dominating set in $\mathcal{D}(G_\infty) \cup \mathcal{D}'(G_\infty)$.*

Proof. In this proof, we are going to demonstrate that any of the four possible attacks (one per unoccupied square) around a vertex in a D_t (or D'_t) placement can be defended by Rotate-Square and, most importantly, the guards still occupy a placement in $\mathcal{D}(G_\infty) \cup \mathcal{D}'(G_\infty)$ after their turn. Below, in Figure 4.8, we provide pictorial details for one out of eight cases, four for D_t and four for D'_t. We need not care about the value of t, since all D_t, respectively D'_t, placements are mere shifts of each other. The defence-responsible guard is shown in black, while the rest is depicted in gray. Also, notice that the pattern guards' cousins occupy positions $(w \pm 5\alpha, z \pm 5\beta)$ for $\alpha, \beta \in \mathbb{N}$, where (w, z) is the new position of a pattern square guard. Then, $f(w, z) = f(w \pm 5\alpha, z \pm 5\beta)$ and $f'(w, z) = f'(w \pm 5\beta, z \pm 5\alpha)$, since the modulo 5 operation cancels out the addition (subtraction) of 5α and 5β. A similar observation holds for stationary guards: we identify a model guard, say on position (a, b), and then the rest of such guards are given by $(a \pm 5\alpha, b \pm 5\beta)$. Again, the $f(\cdot)$, respectively $f'(\cdot)$, values of all these vertices remain equal. For this reason, we focus below only on the pattern guards and one stationary guard and demonstrate that they share the same value of $f(\cdot)$, respectively $f'(\cdot)$. We hereby consider a potential attack around a vertex $(x, y) \in D_t$.

Attack on $(x-1, y)$ (i.e., on $SQ_3(x, y)$). We apply Rotate-Square around $SQ_0(x, y)$. The four pattern guards and the model stationary guard move as follows (Figure 4.8):

Let P stand for the set of new positions given in Table 4.1. The guards now occupy cells $(w, z) \in P$ where $f'(w, z) = 2x + y - 2 \pmod 5 = 2x + y + 3 \pmod 5 = t'$. By this fact, we get $P \subseteq D'_{t'}$. Now, assume there exists a cell $(w, z) \notin P$, but $(w, z) \in D'_{t'}$. Without loss of generality, we assume $w \in [x-3, x+1]$ and $z \in [y-1, y+3]$, since the configuration of the guards in this window is copied all over the grid by the symmetry of D_t or D'_t placements. Since $(w, z) \notin P$, this is a cell with no guard on it. However, by construction, any such cell is dominated by an adjacent vertex (w_1, z_1) with $f'(w_1, z_1) = t'$. Then, by assumption, $f'(w, z) = f'(w_1, z_1) = t'$, which is a contradiction because, by definition of $f'(\cdot)$, two adjacent cells never have equal values.

All other cases are proved in a similar fashion; for the details, see Tables 4.2–4.8. Note that an attack on a D_t placement leads to a $D'_{t'}$ placement for some t' and vice versa. □

Theorem 11. *The guards eternally dominate G_∞ by following the Rotate-Square strategy starting from an initial dominating set in $\mathcal{D}(G_\infty) \cup \mathcal{D}'(G_\infty)$.*

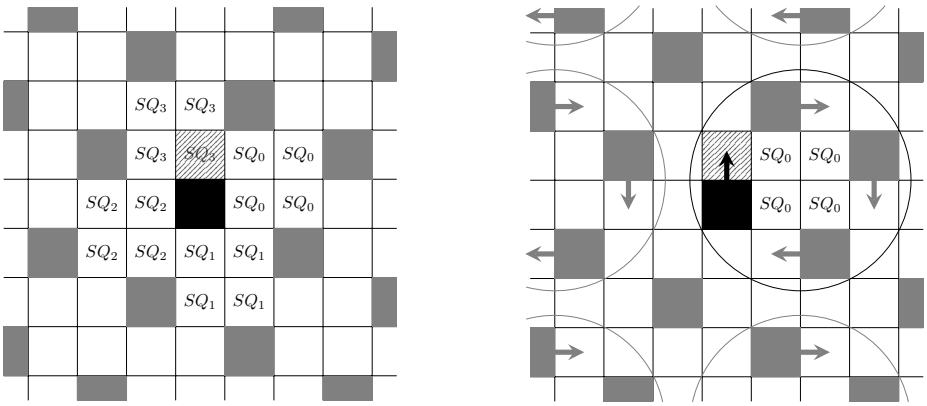
Proof. We prove by induction that the guards defend against any number of attacks and always maintain a placement in $\mathcal{D}(G_\infty) \cup \mathcal{D}'(G_\infty)$ after their turn.

In the first step, the guards apply Rotate-Square and by Lemma 15 they successfully defend against the first attack and now form another dominating set in $\mathcal{D}(G_\infty) \cup \mathcal{D}'(G_\infty)$.

Assume that i attacks have occurred and the guards have successfully defended against all of them by following Rotate-Square. That is, they occupy a configuration in $\mathcal{D}(G_\infty) \cup \mathcal{D}'(G_\infty)$. The $(i+1)$-st attack now occurs: the guards again follow Rotate-Square, defend against the attack and form another dominating set in $\mathcal{D}(G_\infty) \cup \mathcal{D}'(G_\infty)$ (Lemma 15). □

Table 4.1: Attack on $(x-1, y)$ (rotate around $SQ_0(x, y)$); Figure 4.8

Guard	Old Position (w, z)	New Position (w', z')	$f'(w', z')$ (mod 5)
defence-responsible	(x, y)	$(x-1, y)$	$2x + y - 2$
pattern	$(x-2, y+1)$	$(x-2, y+2)$	$2x + y - 2$
pattern	$(x-1, y+3)$	$(x, y+3)$	$2x + y + 3$
pattern	$(x+1, y+2)$	$(x+1, y+1)$	$2x + y + 3$
stationary	$(x-3, y-1)$	$(x-3, y-1)$	$2x + y - 2$

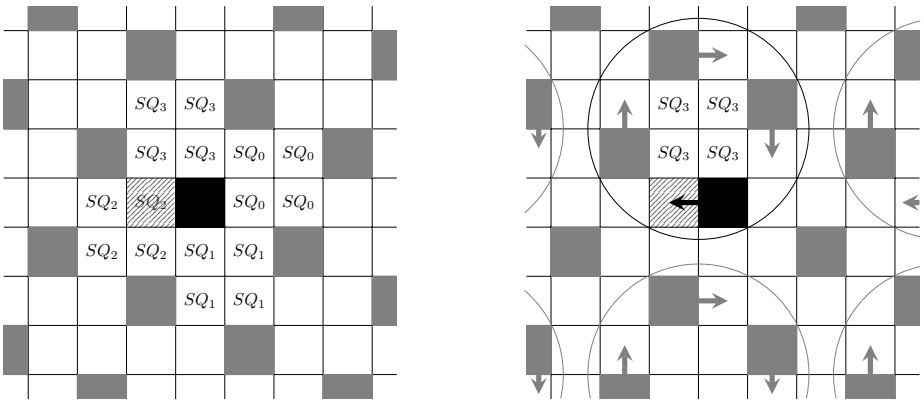


(a) Attack on bottom-right cell of $SQ_3(x, y)$ (b) Rotate-Square: $SQ_0(x, y)$ pattern square

Figure 4.8: Defending against an attack on $SQ_3(x, y)$

Table 4.2: Attack on $(x, y-1)$ (rotate around $SQ_3(x, y)$); Figure 4.9

Guard	Old Position (w, z)	New Position (w', z')	$f'(w', z')$ (mod 5)
defence-responsible	(x, y)	$(x, y-1)$	$2x + y - 1$
pattern	$(x-1, y-2)$	$(x-2, y-2)$	$2x + y - 1$
pattern	$(x-3, y-1)$	$(x-3, y)$	$2x + y - 1$
pattern	$(x-2, y+1)$	$(x-1, y+1)$	$2x + y - 1$
stationary	$(x+1, y+2)$	$(x+1, y+2)$	$2x + y + 4$

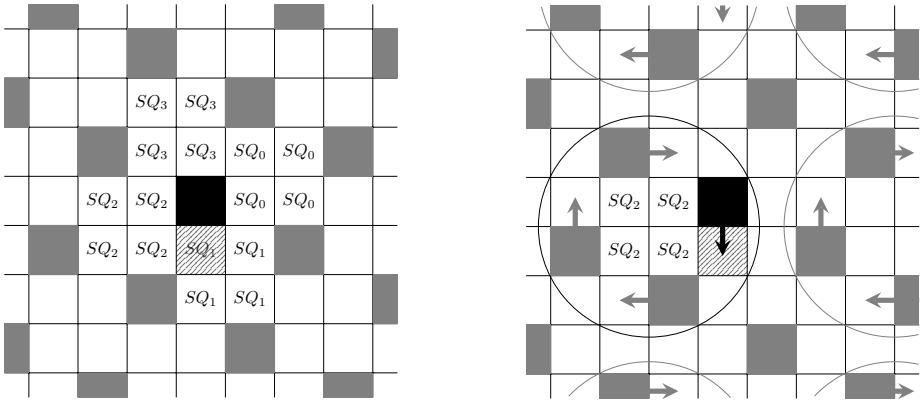


(a) Attack on top-right cell of $SQ_2(x, y)$ (b) Rotate-Square: $SQ_3(x, y)$ pattern square

Figure 4.9: Defending against an attack on $SQ_2(x, y)$

Table 4.3: Attack on $(x+1, y)$ (rotate around $SQ_2(x, y)$); Figure 4.10

Guard	Old Position (w, z)	New Position (w', z')	$f'(w', z')$ (mod 5)
defence-responsible	(x, y)	$(x+1, y)$	$2x+y+2$
pattern	$(x+2, y-1)$	$(x+2, y-2)$	$2x+y+2$
pattern	$(x+1, y-3)$	$(x, y-3)$	$2x+y-3$
pattern	$(x-1, y-2)$	$(x-1, y-1)$	$2x+y-3$
stationary	$(x-2, y+1)$	$(x-2, y+1)$	$2x+y-3$

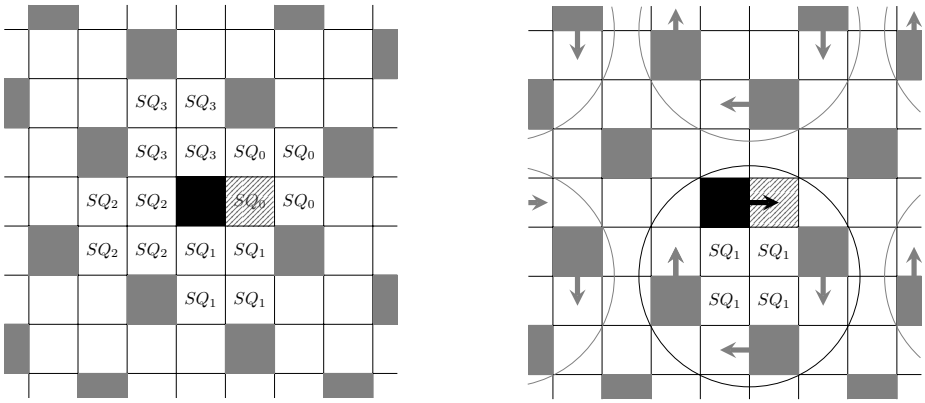


(a) Attack on top-left cell of $SQ_1(x, y)$ (b) Rotate-Square: $SQ_2(x, y)$ pattern square

Figure 4.10: Defending against an attack on $SQ_1(x, y)$

Table 4.4: Attack on $(x, y+1)$ (rotate around $SQ_1(x, y)$); Figure 4.11

Guard	Old Position (w, z)	New Position (w', z')	$f'(w', z')$ (mod 5)
defence-responsible	(x, y)	$(x, y+1)$	$2x+y+1$
pattern	$(x+1, y+2)$	$(x+2, y+2)$	$2x+y+1$
pattern	$(x+3, y+1)$	$(x+3, y)$	$2x+y+1$
pattern	$(x+2, y-1)$	$(x+1, y-1)$	$2x+y+1$
stationary	$(x-1, y+3)$	$(x-1, y+3)$	$2x+y+1$

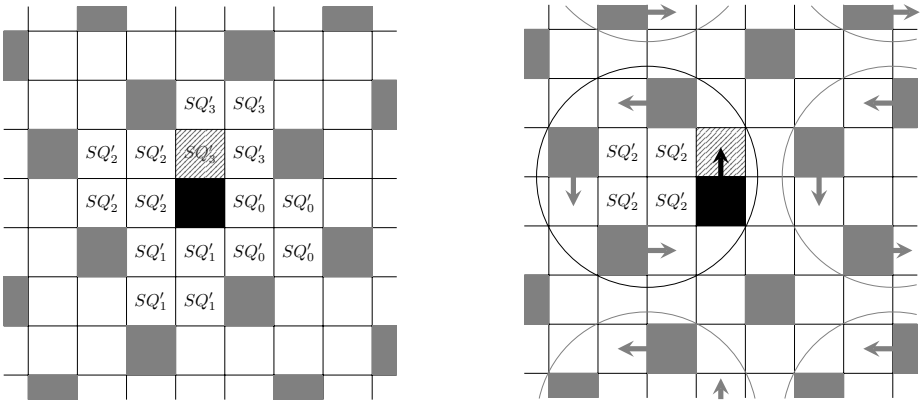


(a) Attack on bottom-left cell of $SQ_0(x, y)$ (b) Rotate-Square: $SQ_1(x, y)$ pattern square

Figure 4.11: Defending against an attack on $SQ_0(x, y)$

Table 4.5: Attack on $(x-1, y)$ (rotate around $SQ'_2(x, y)$); Figure 4.12

Guard	Old Position (w, z)	New Position (w', z')	$f(w', z') \pmod 5$
defence-responsible	(x, y)	$(x-1, y)$	$x+2y-1$
pattern	$(x-2, y-1)$	$(x-2, y-2)$	$x+2y-1$
pattern	$(x-1, y-3)$	$(x, y-3)$	$x+2y-1$
pattern	$(x+1, y-2)$	$(x+1, y-1)$	$x+2y-1$
stationary	$(x-3, y+1)$	$(x-3, y+1)$	$x+2y-1$

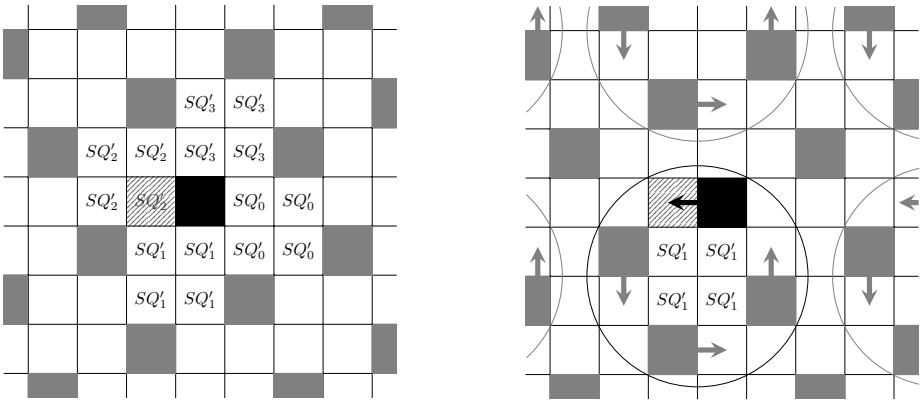


(a) Attack on bottom-left cell of $SQ'_3(x, y)$ (b) Rotate-Square: $SQ'_2(x, y)$ pattern square

Figure 4.12: Defending against an attack on $SQ'_3(x, y)$

Table 4.6: Attack on $(x, y-1)$ (rotate around $SQ'_1(x, y)$); Figure 4.13

Guard	Old Position (w, z)	New Position (w', z')	$f(w', z') \pmod 5$
defence-responsible	(x, y)	$(x, y-1)$	$x+2y-2$
pattern	$(x+2, y+1)$	$(x+1, y+1)$	$x+2y+3$
pattern	$(x+3, y-1)$	$(x+3, y)$	$x+2y+3$
pattern	$(x+1, y-2)$	$(x+2, y-2)$	$x+2y-2$
stationary	$(x-1, y+2)$	$(x-1, y+2)$	$x+2y+3$

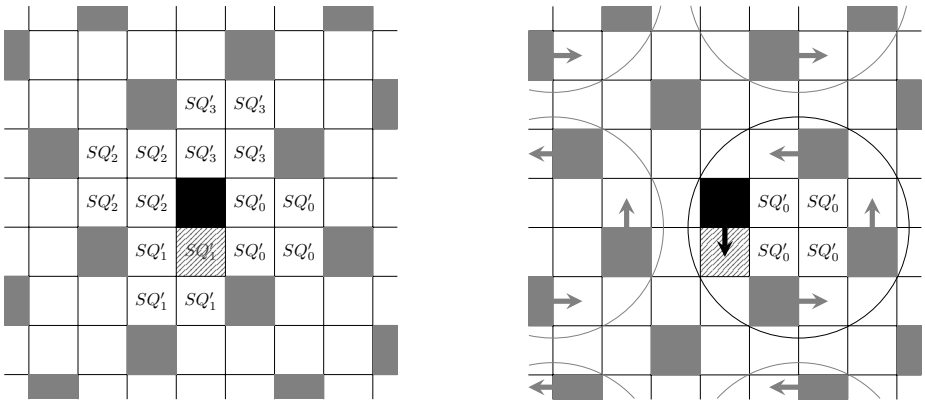


(a) Attack on bottom-right cell of $SQ'_2(x, y)$ (b) Rotate-Square: $SQ'_1(x, y)$ pattern square

Figure 4.13: Defending against an attack on $SQ'_2(x, y)$

Table 4.7: Attack on $(x+1, y)$ (rotate around $SQ'_0(x, y)$); Figure 4.14

Guard	Old Position (w, z)	New Position (w', z')	$f(w', z')$ (mod 5)
defence-responsible	(x, y)	$(x+1, y)$	$x+2y+1$
pattern	$(x+2, y+1)$	$(x+2, y+2)$	$x+2y+1$
pattern	$(x+1, y+3)$	$(x, y+3)$	$x+y+1$
pattern	$(x-1, y+2)$	$(x-1, y+1)$	$x+2y+1$
stationary	$(x-2, y-1)$	$(x-2, y-1)$	$x+2y-4$

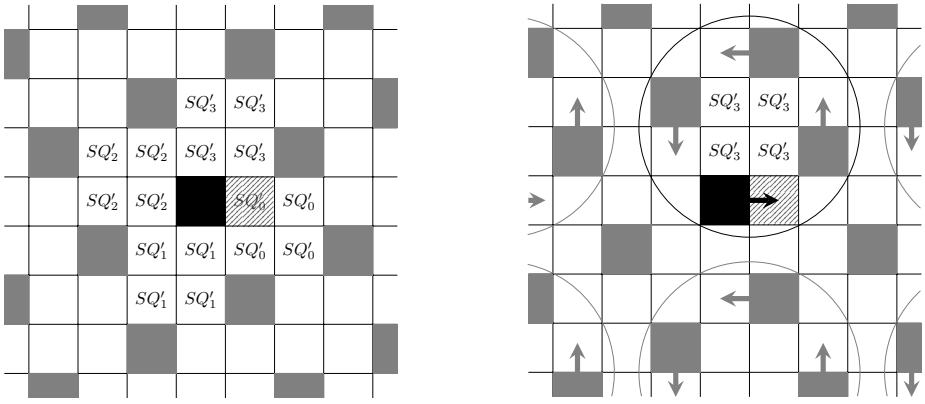


(a) Attack on top-right cell of $SQ'_1(x, y)$ (b) Rotate-Square: $SQ'_0(x, y)$ pattern square

Figure 4.14: Defending against an attack on $SQ'_1(x, y)$

Table 4.8: Attack on $(x, y+1)$ (rotate around $SQ'_3(x, y)$); Figure 4.15

Guard	Old Position (w, z)	New Position (w', z')	$f(w', z')$ (mod 5)
defence-responsible	(x, y)	$(x, y+1)$	$x+2y+2$
pattern	$(x-1, y+2)$	$(x-2, y+2)$	$x+2y+2$
pattern	$(x-3, y+1)$	$(x-3, y)$	$x+2y-3$
pattern	$(x-2, y-1)$	$(x-1, y-1)$	$x+2y+-3$
stationary	$(x+1, y+3)$	$(x+1, y+3)$	$x+2y+2$



(a) Attack on top-left cell of $SQ'_0(x, y)$ (b) Rotate-Square: $SQ'_3(x, y)$ pattern square

Figure 4.15: Defending against an attack on $SQ'_0(x, y)$

4.4 Eternally Dominating Finite Grids

We now apply the Rotate-Square strategy to finite grids, i.e., graphs of the form $P_m \Box P_n$. The initial idea is to follow the rules of the strategy, but to never leave any boundary or corner vertex unoccupied. A finite $m \times n$ grid consists of vertices (i, j) where $i \in \{0, 1, 2, \ldots, m-1\}$ and $j \in \{0, 1, 2, \ldots, n-1\}$. Vertices $(0, x), (m-1, x), (y, 0), (y, n-1)$ for $x \in \{1, 2, \ldots, n-2\}$ and $y \in \{1, 2, \ldots, m-2\}$ are called *boundary vertices*, while vertices $(0, 0), (0, n-1), (m-1, 0), (m-1, n-1)$ are called *corner vertices*. Adjacencies are similar to the infinite grid case. However, boundary vertices only have three neighbors, while corner vertices only have two. Let us consider $V(t) = D_t \cap (P_m \Box P_n)$ and $V'(t) = D'_t \cap (P_m \Box P_n)$, respectively. We cite the following counting lemma from [31].

Lemma 16 (Lemma 2.2 [31])**.** *For all t, it holds $\lfloor \frac{mn}{5} \rfloor \leq |V(t)| \leq \lceil \frac{mn}{5} \rceil$, and there exist t_0, t_1, such that $|V(t_0)| = \lfloor \frac{mn}{5} \rfloor$ and $|V(t_1)| = \lceil \frac{mn}{5} \rceil$.*

The main observation in the proof of the above lemma is that there exist either $\lfloor \frac{m}{5} \rfloor$ or $\lfloor \frac{m}{5} \rfloor + 1$ D_t-vertices in one column of a $P_m \Box P_n$ grid. Then, a case-analysis counting provides the above bounds. The same observation holds for D'_t, since $f'(\cdot)$ is defined based on the same function $f : \mathbb{Z}^2 \to \mathbb{Z}_5$. Thence, we can extend the above lemma for D'_t cases with a similar proof.

Lemma 17. *For all t, it holds $\lfloor \frac{mn}{5} \rfloor \leq |V'(t)| \leq \lceil \frac{mn}{5} \rceil$, and there exist t_0, t_1, such that $|V'(t_0)| = \lfloor \frac{mn}{5} \rfloor$ and $|V'(t_1)| = \lceil \frac{mn}{5} \rceil$.*

In order to study the domination of $P_m \Box P_n$, the analysis is based on examining $V(t)$, but for an extended $P_{m+2} \Box P_{n+2}$ mesh. Indeed, Chang [31] showed the following.

Lemma 18 (Theorem 2.2 [31])**.** *For any $m, n \geq 8$, $\gamma_{m,n} \leq \lfloor \frac{(m+2)(n+2)}{5} \rfloor - 4$.*

The result follows by picking an appropriate D_t placement and forcing into the boundary of $P_m \Box P_n$ the guards on the boundary of $P_{m+2} \Box P_{n+2}$. Moreover, Chang showed how to eliminate another four guards; one near each corner.

Below, to facilitate the readability of our analysis, we focus on a specific subcase of finite grids. We demonstrate an m-eternal dominating strategy for $m \times n$ finite grids where $m \bmod 5 = n \bmod 5 = 2$ and then we improve upon it and extend to the general case.

4.4.1 A First Upper Bound: Full Boundary Cover

Initially, we place our guards on vertices belonging to $V(t) = D_t \cap (P_m \Box P_n)$ for some value $t \in \mathbb{Z}_5$. Unlike the approach in [31], we do not force inside any guards lying outside the boundary of $P_m \Box P_n$. Since a sequence of attacks may force the guards to any $V(t)$ or $V'(t)$ placement, i.e., for any value of t, we pick an initial guard placement, say $V(t_1)$, for which $|V(t_1)| = \lceil \frac{mn}{5} \rceil$ holds, to make sure that there are enough guards to maintain domination

while transitioning from one placement to the next. By Lemma 16, such a placement exists. Moreover, we cover the whole boundary by placing a guard on each unoccupied boundary or corner vertex. For an example, see Figure 4.16: black cells stand for guards which are members of a D_t placement, whereas shaded cells denote the places where the extra guards are put. We refer to each of these added guards as a *boundary guard*. This concludes the initial placement of the guards.

The guards now follow Rotate-Square limited within the grid boundaries. For grid regions lying far from the boundary, Rotate-Square is applied in the same way as in the infinite grid case. For guard moves happening near the boundary or the corners, Rotate-Square's new placement demands can be satisfied by performing *shifts of boundary guards*. In other words, a guard may need to step *out of* the boundary, because he is (a cousin of) a pattern guard. Then, another guard steps *into* the boundary to replace him, while the boundary guards between the into and out-of cells shift one step on the boundary. An example can be found in Figure 4.17a, where we partially view the area near the bottom-left corner of a finite grid. The circles enclose repetitions of the pattern square movement. We focus our attention in the following two cases.

- Guard transitions within the boundary: As indicated in the leftmost column in Figure 4.17a, to follow the pattern move, the two black guards have to move downward. However, since their movement does not force them outside the boundary and the boundary is fully occupied, there is no need to move in this case. We demonstrate this by removing the arrows in Figure 4.17b.

- Guard transitions into and out of the boundary: As indicated in the enclosed rectangle at the bottom-center in Figure 4.17b, following the pattern means a guard has to leave the boundary, whereas another has to enter it. To perform the pattern move, while maintaining a full boundary, we perform a shift of boundary guards as depicted in Figure 4.18. Essentially, the three boundary guards between the two pattern guards shift one step to the left to both cover the unoccupied cell left by the pattern guard leaving the boundary and free a cell for the new position of the pattern guard entering the boundary.

We refer to this slightly modified version of Rotate-Square as *Finite Rotate-Square*.

Lemma 19. *Assume* $m \bmod 5 = n \bmod 5 = 2$ *and that the guards follow Finite Rotate-Square, for an m-Eternal Domination game in* $P_m \square P_n$*. Then, after every turn, their new placement* P *is dominating, all boundary and corner vertices have a guard on them and, for some* $t \in \mathbb{Z}_5$*, there exists a set* $V(t)$ *(or* $V'(t)$*) such that* $V(t) \subseteq P$ *(or* $V'(t) \subseteq P$*).*

Proof. Consider the $(m-2) \times (n-2)$ subgrid that remains when we remove the boundary rows and columns. Since $m \bmod 5 = n \bmod 5 = 2$, $(m-2)$ and $(n-2)$ perfectly divide 5. The latter means that each row, respectively column, of the subgrid has exactly $\frac{n-2}{5}$,

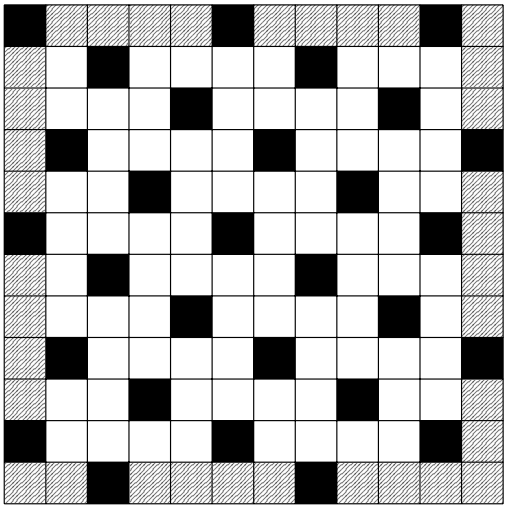
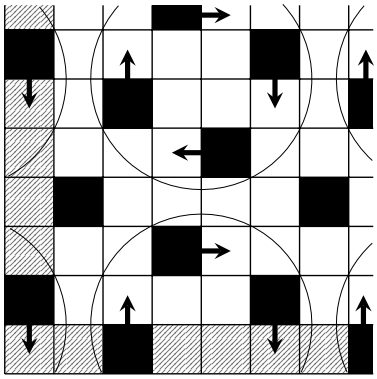
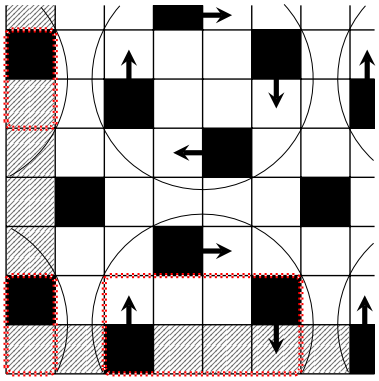


Figure 4.16: An example of an initial D_t placement for the guards in a 12×12 grid: black cells stand for D_t guards, whereas shaded cells stand for extra boundary guards



(a) Circles enclosing pattern square repetitions near a corner of a finite grid



(b) Guard transitions within and into/ out of the boundary (within dashed rectangles)

Figure 4.17: Example of Finite Rotate-Square near the bottom-left corner of a finite grid

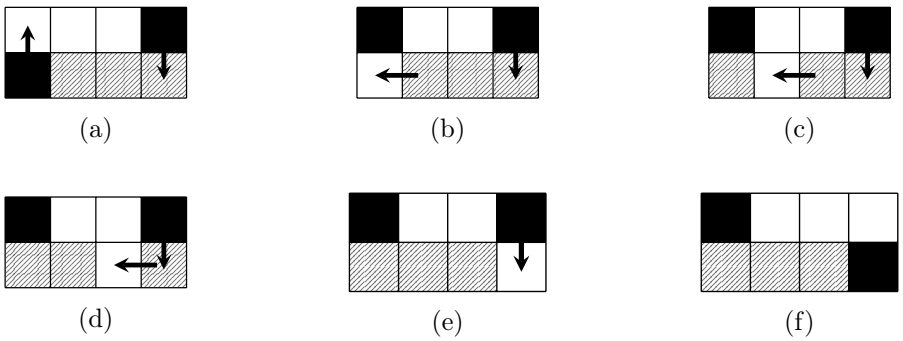
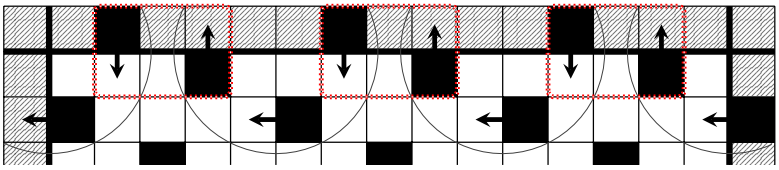
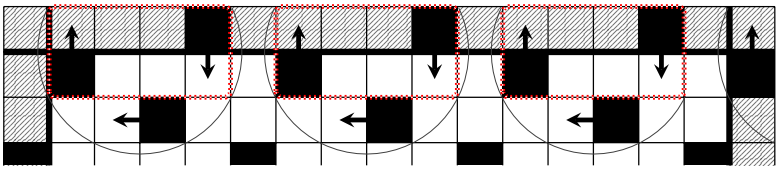


Figure 4.18: An example of boundary guards' shifting for the case of Figure 4.17b



(a) $\frac{n-2}{5} = 3$ pairs: guards of adjacent squares



(b) $\frac{n-2}{5} = 3$ pairs: guards of same square

Figure 4.19: Examples of boundary-shifting pairs on the upper boundary of a grid ($n = 17$)

respectively $\frac{m-2}{5}$, guards on it. Without loss of generality, consider row one neighboring the upper boundary row, which is row zero. Let us assume that a pattern square propagation forces a row-one guard to move into the boundary. Then, by symmetry of the pattern guards' move, there exists another guard on the boundary row zero that needs to move downward to row one. Notice that the same holds for each of the $\frac{n-2}{5}$ guards lying on row one, since the pattern guards' move propagates in hops of distance five. Movements in and out of the boundary alternate due to the shape of the pattern square. Moreover, we do not need to care about where the pattern square repetition is "cut" by the left/right boundary since, due to $n-2$ perfectly dividing 5, there are exactly $\frac{n-2}{5}$ full pattern squares occurring subject to shifting. Consequently, we can apply the shifting procedure demonstrated in Figure 4.18 to apply the moves and maintain a full boundary, while preserving the number of guards on row one. In Figure 4.19, we demonstrate examples of the above remarks: $\frac{n-2}{5}$ guards move from row one into row zero, whereas $\frac{n-2}{5}$ move vice versa. By the discussion above, it is always possible to form pairs of leaving/entering boundary guards and apply the shifting procedure demonstrated in Figure 4.18 either leftward or rightward.

The new placement P is dominating, since the $(m-2) \times (n-2)$ subgrid is dominated by any $V(t)$ or $V'(t)$ placement and the boundary is always full of guards. Moreover, since we follow a modified Rotate-Square, P contains as a subset a vertex set $V(t)$ or $V'(t)$ after each guards' turn. □

Lemma 20. *For $m, n \geq 7$ such that $m \bmod 5 = n \bmod 5 = 2$, $\gamma^{\infty}_{m,n} \leq \frac{mn}{5} + \frac{8}{5}(m+n) - \frac{16}{5}$ holds.*

Proof. By inductive application of Lemma 19, Finite Rotate-Square eternally dominates $P_m \square P_n$.

From the initial $V(t)$ placement, we get exactly $\frac{(m-2)(n-2)}{5}$ guards within $P_{m-2} \square P_{n-2}$, since $(m-2)$ and $(n-2)$ perfectly divide 5. Then, we need another $2(m+n)-4$ guards

to cover the whole boundary. Overall, the guards sum to $\frac{(m-2)(n-2)}{5} + 2(m+n) - 4 = \frac{mn}{5} + \frac{8}{5}(m+n) - \frac{16}{5}$. □

4.4.2 An Improved Upper Bound: Partial Boundary Cover

In the version of Finite Rotate-Square just presented above, the entirety of the boundary always remains covered. More specifically, five guards are placed for every sequence of five non-corner boundary vertices. Optimistically, we would like to lower the number of guards to two guards per every five boundary vertices. Then, compared to the standard domination number, this would provide only a constant-factor additive term. In this subsection, we prove an improved upper bound for Finite Rotate-Square by using three guards for each sequence of five non-corner boundary vertices. Furthermore, we discuss why having two guards would instead most likely fail for Finite Rotate-Square (or simple variations of it).

Lemma 21. *For $m, n \geq 7$ such that $m \bmod 5 = n \bmod 5 = 2$, $\gamma^{\infty}_{m,n} \leq \frac{mn}{5} + \frac{4}{5}(m+n)$ holds.*

Proof. First, let us take advantage of the condition $m \bmod 5 = n \bmod 5 = 2$ in order to reduce this family of grids to the 7×7 grid case. Imagine a general $m \times n$ grid where $m \bmod 5 = n \bmod 5 = 2$ holds. The non-boundary vertices can be partitioned into $\frac{(m-2)(n-2)}{5}$ subgrids of size 5×5, e.g., see Figure 4.20. Moreover, we add four guards, one in each corner, which never move throughout the execution of the strategy, since they can never leave the boundary. Then, we can partition each boundary row/column of the grid into sequences of length five.

Now, notice that the far-from-the-boundary guards implement Rotate-Square and, due to the modulo 5 use in the emergent D_t and D'_t placements, all 5×5 subgrids are copies of each other. Moreover, for the same reason, all segments of length five on the same boundary row/column look identical at all times. Thence, we can contract all 5×5 subgrids and side segments until a 7×7 grid is left. Below, we provide a strategy for this special case. For $m, n > 7$, the strategy can be extracted by copying the 7×7 strategy in each subgrid and boundary row/column segment.

Hence, to prove the bound, it suffices to provide an m-eternal domination strategy for the 7×7 grid with each corner always occupied by an immovable guard and three guards guarding each boundary row/column of length five. In Figure 4.21, we demonstrate such placements for the 7×7 grid: in column A on the left, we depict all possible D_t placements with the corresponding boundary cover, whereas, in column B on the right, we depict all possible D'_t placements. Moreover, in column A, we provide all the guard transitions for an attack to an unoccupied vertex. Transitions in column B are omitted since all guard movements are reversible.

By inductively applying the strategy of Figure 4.21, this improved version of Finite Rotate-Square eternally dominates $P_m \Box P_n$, since the guards always form an A or B

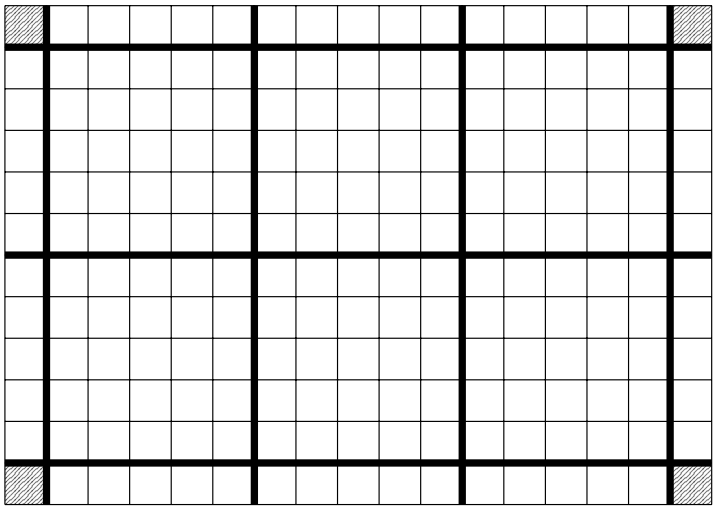


Figure 4.20: An example of partitioning a 17×12 grid into 5×5 subgrids

placement. From the initial $V(t)$ placement, we get exactly $\frac{(m-2)(n-2)}{5}$ guards within $P_{m-2} \square P_{n-2}$, since $(m-2)$ and $(n-2)$ perfectly divide 5. Then, we require four guards for the corners and another $\frac{3}{5}(m-2)$, respectively $\frac{3}{5}(n-2)$, to cover a side of the grid. Overall, we get $\frac{(m-2)(n-2)}{5} + 2 \cdot \frac{3}{5}(m-2+n-2) + 4 = \frac{mn}{5} + \frac{4}{5}(m+n)$ guards suffice for m-eternal domination. □

The above proof is crucially based on the fact that $\gamma_{\mathrm{m}}^{\infty}(P_5) = 3$. Indeed, it is easy to verify that two guards cannot m-eternally dominate a path of length five. Therefore, a uniform approach as the one taken in the proof of Lemma 21 is bound to fail. Furthermore, for non-uniform boundary guarding approaches, the problem seems to persist. In such approaches, boundary guards are not dedicated to a single P_5 segment of the side. Intuitively, the latter can easily lead to the creation of bias, meaning that eventually the extra corner guard (or any constant number of extra corner guards) has to move in order to assist with the protection of the boundary and leave the corner unoccupied.

4.4.3 A General Upper Bound

So far, we have focused on the special case $m \bmod 5 = n \bmod 5 = 2$ for which we provided an upper bound for the m-eternal domination number. We generalize this bound for arbitrary m, n values.

Lemma 22. *For $m, n \geq 7$, $\gamma_{m,n}^{\infty} \leq \frac{mn}{5} + \mathcal{O}(m+n)$ holds.*

Proof. The idea behind this general bound is to "thicken" the boundary in the cases when $m \bmod 5 = n \bmod 5 = 2$ does not hold and then apply Finite Rotate-Square as above. More formally, one can identify an $(m-i) \times (n-j)$ subgrid, in the interior of the $m \times n$ grid, where $i, j \leq 5$ such that $(m-i) \bmod 5 = (n-j) \bmod 5 = 2$ and execute the strategy

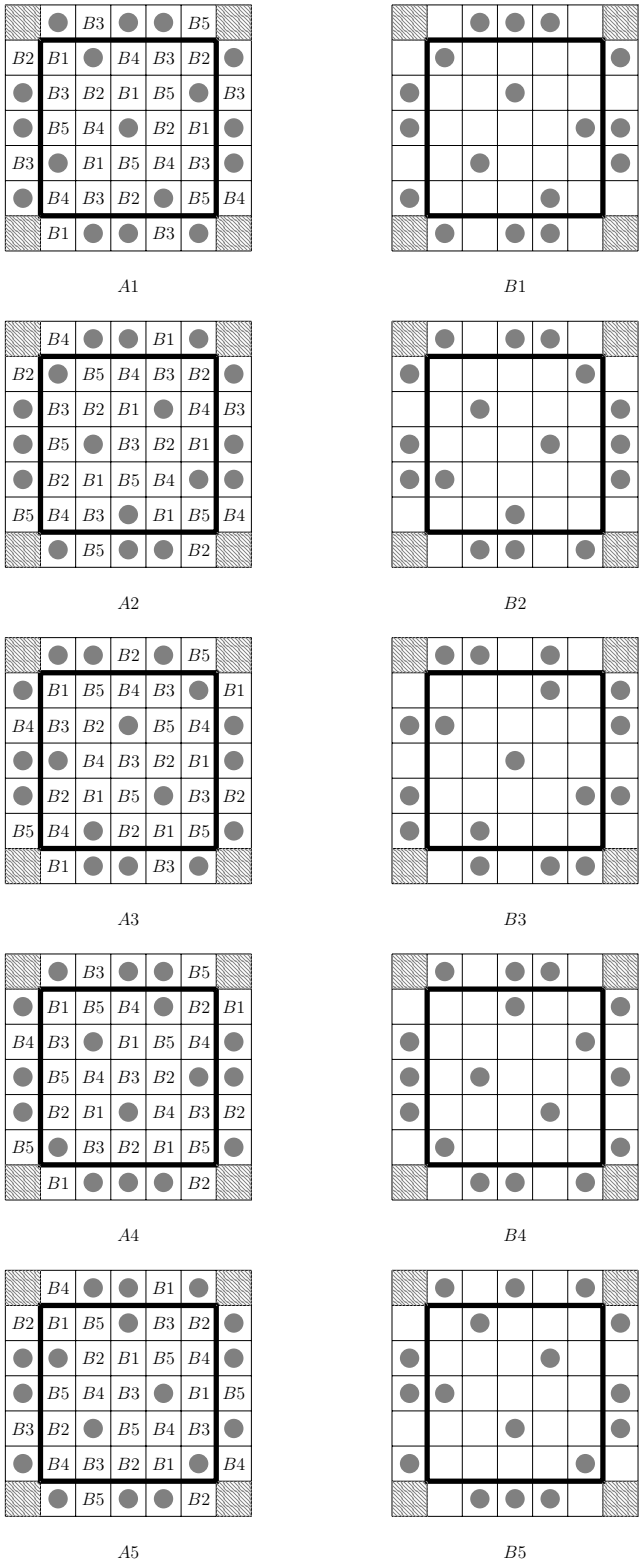


Figure 4.21: Improved Finite Rotate-Square (reduced to the 7×7 grid)

there. The rest of the rows and columns can be eternally secured by populating them with $\mathcal{O}(m+n)$ extra guards, e.g, in the worst-case, place one guard at each such cell. □

Gonçalves et al. [67] showed $\gamma_{m,n} \geq \lfloor \frac{(m+2)(n+2)}{5} \rfloor - 4$ for any $m, n \geq 16$. By combining this with Lemma 18, we get the exact domination number $\gamma_{m,n} = \lfloor \frac{(m+2)(n+2)}{5} \rfloor - 4$ for $m, n \geq 16$. Then, by using Lemma 22, our main result follows.

Theorem 12. *For any $m, n \geq 16$, $\gamma^{\infty}_{m,n} \leq \gamma_{m,n} + \mathcal{O}(m+n)$ holds.*

4.5 Concluding Remarks

We demonstrated a first strategy to m-eternally dominate general rectangular grids based on the repetition of a rotation pattern. Regarding further work, a more careful case analysis of the boundary may lead to improvements regarding the coefficient of the linear term. It is unclear whether this strategy can be used to obtain a constant additive gap between the domination and the m-eternal domination number in large grids. Furthermore, the existence of a stronger lower bound than the trivial $\gamma^{\infty}_{m,n} \geq \gamma_{m,n}$ bound also remains open.

With respect to future research directions, one could consider multiple attacks to address more realistic intrusion scenarios. Also, being able to protect from a distance $k > 1$, i.e., giving more power to the guards, might have strong correlations to the $k = 1$ case we consider here. Finally, it would be interesting to consider how the centralized graph-theoretic approaches could be moved over to a more distributed setting where the guards only have limited capabilities and must make decisions locally.

Chapter 5

Conclusions

Within this thesis, we researched three different problems with their common characteristic being that of agent mobility. Looking at search problems through the different lenses of Distributed Search and Combinatorial Games, we designed and analyzed strategies for a set of agents to evacuate (Chapter 2), explore (Chapter 3) or defend (Chapter 4) their environment effectively. The problems we study provide a small, yet indicative, flavor of the potential applications in token navigation. We expect the evergrowing improvements in technology to provide novel motivation in this setting. As an example, consider the recent developments in soft-material robotics [111] which could lead to more adaptive robot navigation in complex environments.

One general future work direction is to move away from the centralized point of view and consider the same problems, or other problems of similar nature, in a more distributed fashion. For example, one could include more robots. Additionally, model robots with more limited communication or memory capabilities. Some features of significant research potential are environment dynamics, energy, and robustness.

Interest in dynamic graphs, otherwise called temporal networks [96], has rekindled over the last decade, as they are able to model several aspects of everyday interactions. Designing and analyzing the movement of agents in such disorderly-changing environments may prove to be a task of notable difficulty, e.g., see [51, 97] for two exploration examples.

On a different topic, as we are now approaching an era of environmental awareness, studying mobility problems under energy constraints has already attracted attention among researchers, e.g., see [10, 29]. The robots are now perceived as having a battery, or being in need of recharge, and the objective is to minimize the energy consumption besides other traditional measures like the running time or space.

Another noteworthy aspect is that of robustness (or resilience) in navigation: how do small perturbations to an agent's input, or to the environment, affect the performance of the designed strategies? Lower bounds might either be robust, i.e., capture an inherent difficulty of the problem in question, or fragile, that is, they only depend on a very specific execution instance. In this respect, a pertinent example is that of smoothed analysis [49, 106].

Overall, this is an exciting research area with an inherent algorithmic element where a variety of mathematical tools apply. Mathematical, as well as real-life, motivation is abundant and therefore we expect the research community around mobility problems to thrive in the future.

References

[1] Martin Aigner and Michael Fromme. A game of cops and robbers. *Discrete Applied Mathematics*, 8(1):1–12, 1984.

[2] Michael Albert, Richard Nowakowski, and David Wolfe. *Lessons in play: an introduction to combinatorial game theory*. CRC Press, 2007.

[3] Romas Aleliunas, Richard M Karp, Richard J Lipton, László Lovász, and Charles Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *20th Annual Symposium on Foundations of Computer Science*, pages 218–223. IEEE, 1979.

[4] Steve Alpern, Robbert Fokkink, L Gąsieniec, Roy Lindelauf, and VS Subrahmanian. *Search theory*. Springer, 2013.

[5] Steve Alpern and Shmuel Gal. *The theory of search games and rendezvous*, volume 55. Springer Science & Business Media, 2006.

[6] Mark Anderson, Christian Barrientos, Robert C. Brigham, Julie Carrington, Richard Vitray, and Jay Yellen. Maximum-demand graphs for eternal security. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 61:111–128, 2007.

[7] Chen Avin, Michal Koucký, and Zvi Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *International Colloquium on Automata, Languages, and Programming*, pages 121–132. Springer, 2008.

[8] Ricardo A Baeza-Yates, Joseph C Culberson, and Gregory JE Rawlins. Searching in the plane. *Information and computation*, 106(2):234–252, 1993.

[9] Judit Bar-Ilan and Dror Zernik. Random leaders and random spanning trees. In *International Workshop on Distributed Algorithms*, pages 1–12. Springer, 1989.

[10] Andreas Bärtschi, Daniel Graf, and Matús Mihalák. Collective Fast Delivery by Energy-Efficient Agents. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 56:1–56:16, 2018.

[11] Hervé Baumann, Pierluigi Crescenzi, and Pierre Fraigniaud. Parsimonious flooding in dynamic graphs. *Distributed Computing*, 24(1):31–44, 2011.

[12] I Beaton, S Finbow, and J.A. MacDonald. Eternal domination numbers of 4×n grid graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 85:33–48, 2013.

[13] Anatole Beck. On the linear search problem. *Israel Journal of Mathematics*, 2(4):221–228, 1964.

[14] Richard Bellman. Minimization problem. *Bull. Amer. Math. Soc*, 62(3):270, 1956.

[15] Richard Bellman. An optimal search. *Siam Review*, 5(3):274, 1963.

[16] Michael A Bender, Antonio Fernández, Dana Ron, Amit Sahai, and Salil Vadhan. The power of a pebble: Exploring and mapping directed graphs. In *Proceedings of the 30th annual ACM Symposium on Theory of Computing*, pages 269–278. ACM, 1998.

[17] John Bird. *Electrical circuit theory and technology*. Routledge, 2017.

[18] Ralph Boas. Littlewood's miscellany. *American Mathematical Monthly*, 96(2):167–169, 1989.

[19] Anthony Bonato. *The game of cops and robbers on graphs*. American Math. Soc., 2011.

[20] Anthony Bonato, Paweł Prałat, and Changping Wang. Pursuit-evasion in models of complex networks. *Internet Mathematics*, 4(4):419–436, 2007.

[21] A. Bondy and U.S.R. Murty. *Graph Theory*. Graduate Texts in Mathematics. Springer London, 2007.

[22] Prosenjit Bose, Jean-Lou De Carufel, and Stephane Durocher. Revisiting the problem of searching on a line. In *European Symposium on Algorithms*, pages 205–216. Springer, 2013.

[23] Sebastian Brandt, Felix Laufenberg, Yuezhou Lv, David Stolz, and Roger Wattenhofer. Collaboration without communication: Evacuating two robots from a disk. In *International Conference on Algorithms and Complexity*, pages 104–115. Springer, 2017.

[24] Richard L Breisch. *Lost in a Cave: applying graph theory to cave exploration*. National Speological Society, 2012.

[25] Graham Brightwell and Peter Winkler. Maximum hitting time for random walks on graphs. *Random Structures & Algorithms*, 1(3):263–276, 1990.

[26] Marc Bui, Thibault Bernard, Devan Sohier, and Alain Bui. Random walks in distributed computing: A survey. In *International Workshop on Innovative Internet Community Systems*, pages 1–14. Springer, 2004.

[27] AP Burger, EJ Cockayne, WR Grundlingh, CM Mynhardt, JH Van Vuuren, and W Winterbach. Finite order domination in graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 49:159–176, 2004.

[28] AP Burger, EJ Cockayne, WR Grundlingh, CM Mynhardt, JH Van Vuuren, and W Winterbach. Infinite order domination in graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 50:179–194, 2004.

[29] Jérémie Chalopin, Shantanu Das, Matúš Mihal'ák, Paolo Penna, and Peter Widmayer. Data delivery by energy-constrained mobile agents. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pages 111–122. Springer, 2013.

[30] Ashok K Chandra, Prabhakar Raghavan, Walter L Ruzzo, Roman Smolensky, and Prasoon Tiwari. The electrical resistance of a graph captures its commute and cover times. *Computational Complexity*, 6(4):312–340, 1996.

[31] T.Y. Chang. *Domination numbers of grid graphs*. PhD thesis, University of South Florida, 1992.

[32] Yann Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2004)*, pages 302–308. IEEE, 2004.

[33] Marek Chrobak, Leszek Gąsieniec, Thomas Gorry, and Russell Martin. Group search on the line. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 164–176. Springer, 2015.

[34] Huda Chuangpishit, Konstantinos Georgiou, and Preeti Sharma. Average case-worst case tradeoffs for evacuating 2 robots from the disk in the face-to-face model. *arXiv preprint arXiv:1807.08640*, 2018.

[35] Timothy H Chung, Geoffrey A Hollinger, and Volkan Isler. Search and pursuit-evasion in mobile robotics. *Autonomous robots*, 31(4):299, 2011.

[36] Andrea EF Clementi, Claudio Macci, Angelo Monti, Francesco Pasquale, and Riccardo Silvestri. Flooding time of edge-markovian evolving graphs. *SIAM Journal on Discrete Mathematics*, 24(4):1694–1712, 2010.

[37] Andrea EF Clementi, Francesco Pasquale, Angelo Monti, and Riccardo Silvestri. Communication in dynamic radio networks. In *Proceedings of the 26th annual ACM symposium on Principles of Distributed Computing*, pages 205–214. ACM, 2007.

[38] Andrea EF Clementi, Francesco Pasquale, Angelo Monti, and Riccardo Silvestri. Information spreading in stationary markovian evolving graphs. In *IEEE International Symposium on Parallel & Distributed Processing (IPDPS 2009)*, pages 1–12. IEEE, 2009.

[39] Nathann Cohen, Nícolas A. Martins, Fionn Mc Inerney, Nicolas Nisse, Stéphane Pérennes, and Rudini Sampaio. Spy-game on graphs: Complexity and simple topologies. *Theoretical Computer Science*, 725:1 – 15, 2018.

[40] Nathann Cohen, Fionn Mc Inerney, Nicolas Nisse, and Stéphane Pérennes. Study of a combinatorial game in graphs through linear programming. *Algorithmica*, 2018.

[41] J. Czyzowicz, E. Kranakis, D. Krizanc, L. Narayanan, J. Opatrny, and S. Shende. Wireless autonomous robot evacuation from equilateral triangles and squares. In *Ad-hoc, Mobile, and Wireless Networks*, pages 181–194, Cham, 2015. Springer International Publishing.

[42] Jurek Czyzowicz, Leszek Gąsieniec, Thomas Gorry, Evangelos Kranakis, Russell Martin, and Dominik Pajak. Evacuating robots via unknown exit in a disk. In *International Symposium on Distributed Computing*, pages 122–136. Springer, 2014.

[43] Jurek Czyzowicz, Leszek Gąsieniec, Adrian Kosowski, and Evangelos Kranakis. Boundary patrolling by mobile agents with distinct maximal speeds. In *European Symposium on Algorithms*, pages 701–712. Springer, 2011.

[44] Jurek Czyzowicz, Konstantinos Georgiou, Maxime Godon, Evangelos Kranakis, Danny Krizanc, Wojciech Rytter, and Michał Włodarczyk. Evacuation from a disc in the presence of a faulty robot. In *International Colloquium on Structural Information and Communication Complexity*, pages 158–173. Springer, 2017.

[45] Jurek Czyzowicz, Konstantinos Georgiou, Ryan Killick, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, Jaroslav Opatrny, and S Shende. Priority evacuation from a disk using mobile robots. *arXiv preprint arXiv:1805.03568*, 2018.

[46] Jurek Czyzowicz, Konstantinos Georgiou, Ryan Killick, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, Jaroslav Opatrny, and Sunil Shende. God save the queen. *arXiv preprint arXiv:1804.06011*, 2018.

[47] Jurek Czyzowicz, Konstantinos Georgiou, Evangelos Kranakis, Lata Narayanan, Jaroslav Opatrny, and Birgit Vogtenhuber. Evacuating robots from a disk using face-to-face communication. In *International Conference on Algorithms and Complexity*, pages 140–152. Springer, 2015.

[48] Jean-Charles Delvenne, Renaud Lambiotte, and Luis EC Rocha. Diffusion on networked systems is a question of time or structure. *Nature communications*, 6:7366, 2015.

[49] Michael Dinitz, Jeremy T Fineman, Seth Gilbert, and Calvin Newport. Smoothed analysis of dynamic networks. *Distributed Computing*, 31(4):273–287, 2018.

[50] Peter G Doyle and J Laurie Snell. Random walks and electric networks. *arXiv preprint math/0001057*, 2000.

[51] Thomas Erlebach, Michael Hoffmann, and Frank Kammer. On temporal graph exploration. In *International Colloquium on Automata, Languages, and Programming*, pages 444–455. Springer Berlin Heidelberg, 2015.

[52] Uriel Feige. A tight upper bound on the cover time for random walks on graphs. *Random Structures & Algorithms*, 6(1):51–54, 1995.

[53] Ofer Feinerman, Amos Korman, Shay Kutten, and Yoav Rodeh. Fast rendezvous on a cycle by agents with different speeds. In *International Conference on Distributed Computing and Networking*, pages 1–13. Springer, 2014.

[54] Daniel Figueiredo, Philippe Nain, Bruno Ribeiro, Edmundo de Souza e Silva, and Don Towsley. Characterizing continuous time random walks on time varying graphs. In *ACM SIGMETRICS Performance Evaluation Review*, volume 40, pages 307–318. ACM, 2012.

[55] S Finbow, ME Messinger, and M van Bommel. Eternal domination of 3×n grid graphs. *Aust. J. of Combin.*, 61(2):156–174, 2015.

[56] Stephen Finbow, Serge Gaspers, Margaret-Ellen Messinger, and Paul Ottaway. A note on the eternal dominating set problem. *International Journal of Game Theory*, 47(2):543–555, 2018.

[57] Stephen Finbow and Gary MacGillivray. The firefighter problem: a survey of results, directions and questions. *Australasian Journal of Combinatorics*, 43(57-77):6, 2009.

[58] Fedor V Fomin, Serge Gaspers, Petr A Golovach, Dieter Kratsch, and Saket Saurabh. Parameterized algorithm for eternal vertex cover. *Information Processing Letters*, 110(16):702–706, 2010.

[59] Fedor V Fomin, Frédéric Giroire, Alain Jean-Marie, Dorian Mazauric, and Nicolas Nisse. To satisfy impatient web surfers is hard. *Theoretical Computer Science*, 526:1–17, 2014.

[60] Fedor V Fomin and Dimitrios M Thilikos. An annotated bibliography on guaranteed graph searching. *Theoretical computer science*, 399(3):236–245, 2008.

[61] Konstantinos Georgiou, George Karakostas, and Evangelos Kranakis. Search-and-fetch with 2 robots on a disk: Wireless and face-to-face communication models. *arXiv preprint arXiv:1611.10208*, 2016.

[62] Konstantinos Georgiou, George Karakostas, and Evangelos Kranakis. Search-and-fetch with one robot on a disk. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pages 80–94. Springer, 2016.

[63] Frédéric Giroire, Ioannis Lamprou, Dorian Mazauric, Nicolas Nisse, Stéphane Pérennes, and Ronan Soares. Connected surveillance game. *Theoretical Computer Science*, 584:131–143, 2015.

[64] Brian Gluss. An alternative solution to the "lost at sea" problem. *Naval Research Logistics Quarterly*, 8(1):117–122, 1961.

[65] Wayne Goddard, Sandra M Hedetniemi, and Stephen T Hedetniemi. Eternal security in graphs. *J. Combin. Math. Combin. Comput*, 52:169–180, 2005.

[66] John L Goldwasser and William F Klostermeyer. Tight bounds for eternal dominating sets in graphs. *Discrete Mathematics*, 308(12):2589–2593, 2008.

[67] Daniel Gonçalves, Alexandre Pinlou, Michaël Rao, and Stéphan Thomassé. The domination number of grids. *SIAM Journal on Discrete Mathematics*, 25(3):1443–1453, 2011.

[68] Michel Habib, Colin McDiarmid, Jorge Ramirez-Alfonsin, and Bruce Reed. *Probabilistic methods for algorithmic discrete mathematics*, volume 16. Springer Science & Business Media, 2013.

[69] Michael Henning, William Klostermeyer, and Gary MacGillivray. Bounds for the m-eternal domination number of a graph. *Contributions to Discrete Mathematics*, 12(2), 2017.

[70] Michael A Henning and Stephen T Hedetniemi. Defending the roman empire-a new strategy. *Discrete Mathematics*, 266(1-3):239–251, 2003.

[71] Michael A Henning and William F Klostermeyer. Trees with large m-eternal domination number. *Discrete Applied Mathematics*, 211:79–85, 2016.

[72] Till Hoffmann, Mason A Porter, and Renaud Lambiotte. Random walks on stochastic temporal networks. In *Temporal Networks*, pages 295–313. Springer, 2013.

[73] Petter Holme and Jari Saramäki. Temporal networks. *Physics reports*, 519(3):97–125, 2012.

[74] Evan Huus and Evangelos Kranakis. Rendezvous of many agents with different speeds in a cycle. In *International Conference on Ad-Hoc Networks and Wireless*, pages 195–209. Springer, 2015.

[75] Volkan Isler and Nikhil Karnad. The role of information in the cop-robber game. *Theoretical Computer Science*, 399(3):179–190, 2008.

[76] Artur Jeż and Jakub Łopuszański. On the two-dimensional cow search problem. *Information Processing Letters*, 109(11):543–547, 2009.

[77] George Kantor, Sanjiv Singh, Ronald Peterson, Daniela Rus, Aveek Das, Vijay Kumar, Guilherme Pereira, and John Spletzer. Distributed search and rescue with robot and sensor teams. In *Field and Service Robotics*, pages 529–538. Springer, 2003.

[78] Ming-Yang Kao, John H Reif, and Stephen R Tate. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. *Information and Computation*, 131(1):63–79, 1996.

[79] William Klostermeyer, Margaret-Ellen Messinger, and Alejandro Angeli Ayello. An eternal domination problem in grids. *Theory and Applications of Graphs*, 4(1):2, 2017.

[80] William F Klostermeyer and Christina M Mynhardt. Edge protection in graphs. *Australasian J. Combinatorics*, 45:235–250, 2009.

[81] William F Klostermeyer and Christina M Mynhardt. Graphs with equal eternal vertex cover and eternal domination numbers. *Discrete Mathematics*, 311(14):1371–1379, 2011.

[82] William F Klostermeyer and Christina M Mynhardt. Vertex covers and eternal dominating sets. *Discrete Applied Mathematics*, 160(7-8):1183–1190, 2012.

[83] William F Klostermeyer and CM Mynhardt. Protecting a graph with mobile guards. *Applicable Analysis and Discrete Mathematics*, 10:1–29, 2016.

[84] Evangelos Kranakis, Danny Krizanc, and Euripides Markou. The mobile agent rendezvous problem in the ring. *Synthesis Lectures on Distributed Computing Theory*, 1(1):1–122, 2010.

[85] Evangelos Kranakis, Danny Krizanc, and Sergio Rajsbaum. Mobile agent rendezvous: A survey. In *International Colloquium on Structural Information and Communication Complexity*, pages 1–9. Springer, 2006.

[86] Ioannis Lamprou, Russell Martin, and Sven Schewe. Fast two-robot disk evacuation with wireless communication. In *International Symposium on Distributed Computing*, pages 1–15. Springer, 2016.

[87] Ioannis Lamprou, Russell Martin, and Sven Schewe. Perpetually dominating large grids. In *International Conference on Algorithms and Complexity*, pages 393–404. Springer, 2017.

[88] Ioannis Lamprou, Russell Martin, and Sven Schewe. Eternally dominating large grids. *Theoretical Computer Science (in press)*, 2018.

[89] Ioannis Lamprou, Russell Martin, Sven Schewe, Ioannis Sigalas, and Vassilis Zissimopoulos. Maximum rooted connected expansion. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018*, pages 25:1–25:14, 2018.

[90] Ioannis Lamprou, Russell Martin, and Paul Spirakis. Cover time in edge-uniform stochastically-evolving graphs. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 441–455. Springer, 2017.

[91] Ioannis Lamprou, Russell Martin, and Paul Spirakis. Cover time in edge-uniform stochastically-evolving graphs. *Algorithms*, 11(10), 2018.

[92] Danny B Lange and Mitsuru Oshima. Seven good reasons for mobile agents. *Communications of the ACM*, 42(3):88–89, 1999.

[93] Linyuan Lu and Xing Peng. On Meyniel's conjecture of the cop number. *Journal of Graph Theory*, 71(2):192–205, 2012.

[94] Nimrod Megiddo, S Louis Hakimi, Michael R Garey, David S Johnson, and Christos H Papadimitriou. The complexity of searching a graph. *JACM*, 35(1):18–44, 1988.

[95] M. E. Messinger and A. Z. Delaney. Closing the gap: Eternal domination on 3 × n grids. *Contributions to Discrete Mathematics*, 12(1), 2017.

[96] Othon Michail. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics*, 12(4):239–280, 2016.

[97] Othon Michail and Paul G. Spirakis. Traveling salesman problems in temporal graphs. *Theoretical Computer Science*, 634:1–23, 2016.

[98] James R Norris. *Markov chains*. Number 2. Cambridge university press, 1998.

[99] Richard Nowakowski and Peter Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2-3):235–239, 1983.

[100] Torrence D Parsons. Pursuit-evasion in a graph. In *Theory and applications of graphs*, pages 426–441. Springer, 1978.

[101] Debasish Pattanayak, H Ramesh, Partha Sarathi Mandal, and Stefan Schmid. Evacuating two robots from two unknown exits on the perimeter of a disk with wireless communication. In *Proceedings of the 19th International Conference on Distributed Computing and Networking*, number 20. ACM, 2018.

[102] A. Quillot. Jeux et pointes fixes sur les graphes, thése de 3éme cycle. *Université de Paris VI*, pages 131–145, 1978.

[103] Charles S ReVelle and Kenneth E Rosing. Defendens imperium romanum: a classical problem in military strategy. *The American Mathematical Monthly*, 107(7):585–594, 2000.

[104] Luis EC Rocha and Naoki Masuda. Random walk centrality for temporal networks. *New Journal of Physics*, 16(6):063023, 2014.

[105] Alex Scott and Benny Sudakov. A bound for the cops and robbers problem. *SIAM Journal on Discrete Mathematics*, 25(3):1438–1442, 2011.

[106] Daniel A Spielman and Shang-Hua Teng. Smoothed analysis: an attempt to explain the behavior of algorithms in practice. *Communications of the ACM*, 52(10):76–84, 2009.

[107] Michele Starnini, Andrea Baronchelli, Alain Barrat, and Romualdo Pastor-Satorras. Random walks on temporal networks. *Physical Review E*, 85(5):056115, 2012.

[108] Ian Stewart. Defend the roman empire! *Scientific American*, 281(6):136–138, 1999.

[109] Christopher M van Bommel and Martin F van Bommel. Eternal domination numbers of 5×n grid graphs. *J. Comb. Math. Comb. Comput*, 97:83–102, 2016.

[110] Abraham Wald. *Sequential analysis*. Courier Corporation, 1973.

[111] Liyu Wang, Surya Girinatha Nurzaman, and Fumiya Iida. Soft-material robotics. *Foundations and Trends in Robotics*, 5(3):191–259, 2017.

[112] Douglas Brent West. *Introduction to graph theory*. Prentice Hall, 2nd edition, 2000.

[113] Yukiko Yamauchi, Tomoko Izumi, and Sayaka Kamei. Mobile agent rendezvous on a probabilistic edge evolving ring. In *2012 Third International Conference on Networking and Computing*, pages 103–112. IEEE, 2012.

[114] Vladimir Yanovski, Israel A Wagner, and Alfred M Bruckstein. A distributed ant algorithm for efficiently patrolling a network. *Algorithmica*, 37(3):165–186, 2003.